Thèse n° 10 111

Low-Rank Tensor Methods for High-Dimensional Problems

Présentée le 31 mars 2023

Faculté des sciences de base
Algorithmes numériques et calcul haute performance - Chaire CADMOS
Programme doctoral en mathématiques

pour l'obtention du grade de Docteur ès Sciences

par

Christoph Max STRÖSSNER

Acceptée sur proposition du jury

Prof. M. Colombo, présidente du jury
Prof. D. Kressner, directeur de thèse
Prof. H. Harbrecht, rapporteur
Prof. A. Uschmajew , rapporteur
Prof. F. Nobile, rapporteur

École
polytechnique
fédérale
de Lausanne

2023

Acknowledgements

First and foremost, I am genuinely grateful to my supervisor Daniel Kressner for his advice, encouragement, and patience during the last years. Daniel provided me this great opportunity to explore the fascinating world of tensors and their decompositions. Giving me the freedom and flexibility to work on different problems and topics, encouraging me to travel to various conferences and summer schools, trusting me to handle things independently, and guiding me professionally through the unusual circumstances during a global pandemic characterizes Daniel's outstanding support and is far from standard.

I thank Helmut Harbrecht, André Uschmajew and Fabio Nobile for being part of my jury and for taking the time to read my thesis. Special thanks go to Maria Colombo for agreeing to preside over my private defense.

Over the last years, I had the pleasure to collaborate with Sergey Dolgov, Virginie Ehrlacher, Jad Dabaghi and Bonan Sun who provided valuable insights and assistance. I would like to particularly express my gratitude to Virginie Ehrlacher for introducing me to self-diffusion matrices during my six-week stay at CIRM in Luminy as part of CEMRACS 2021 and for inviting me to visit École des ponts ParisTech. I would like to thank Nick Trefethen and the Chebfun team giving me the opportunity to contribute to the official Chebfun package. I am grateful for the help of Behnam Hashemi, Nick Hale, Yuji Nakatsukasa and Alex Townsend.

Moreover, I would like to thank my colleagues and those with whom I had many insightful discussions at EPFL; namely Alice, Amal, Axel, Bonan, David, Francesco, Gianluca, Haoze, Hysan, Ivana, Junli, Kathryn, Lana, Margherita, Minhong, Nico, Stefano, Tingting and Yuxin. Special thanks go to Gianluca who helped proof-reading this thesis.

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

Christoph Strössner

Abstract

In this thesis, we propose and analyze novel numerical algorithms for solving three different high-dimensional problems involving tensors. The commonality of these problems is that the tensors can potentially be well approximated in low-rank formats. Identifying and exploiting this low-rank structure allows us to mitigate the curse of dimensionality.

The first problem considered in this thesis is the computation of functional low-rank approximations of multivariate functions defined on tensor product domains. We develop two novel algorithms to compute such approximations by combining tensorized Chebyshev interpolation with low-rank approximations of the coefficient tensor. For most functions, our numerical experiments demonstrate that our algorithms require a lower number of function evaluations and achieve the same approximation error compared to existing methods. In addition, we solve partial differential equations using a novel global spectral method that can potentially be combined with functional low-rank approximations.

The second problem of interest is the solution of multi-marginal optimal transport problems. After adding entropic regularization, these are equivalent to tensor scaling problems that can be solved using the Sinkhorn algorithm. In literature, it has been suggested to accelerate the Sinkhorn algorithm by exploiting either a graphical model structure or a low-rank tensor approximation. We propose to combine these two approaches to accelerate the solution of the tensor scaling problem even further.

The third problem of interest is the computation of the self-diffusion matrix of a tagged particle process defined on a grid. We propose a novel approach based on computing the matrix via solving a high-dimensional tensor-valued optimization problem. We observe numerically that our approach is much less subject to statistical noise compared to classical approaches based on estimating long-time averages of empirical means of deviations of some stochastic processes.

Key words: low-rank approximation, high-dimensional problems, tensor methods, curse of dimensionality, Chebyshev interpolation, functional low-rank approximation, multi-marginal optimal transport, self-diffusion matrix, spectral method

Zusammenfassung

Diese Arbeit befasst sich mit der Entwicklung und Analyse numerischer Algorithmen zur Lösung dreier hochdimensionaler Probleme. Diese haben gemein, dass sie Tensoren beinhalten, die möglicherweise gut in Niedrigrangformaten approximiert werden können. Durch Ausnutzen dieser Niedrigrangstruktur kann der Fluch der Dimensionalität gelindert werden.

Das erste Problem stellt die Approximation multivariater Funktionen, die auf Tensorproduktmengen definiert sind, mithilfe funktionaler Niedrigrangapproximationen dar. Dafür werden zwei neuartige Algorithmen entwickelt, die die gewünschte Approximation durch Kombination von tensorisierter Tschebyscheff-Interpolation und Niedrigrangapproximation des Koeffiziententensors berechnen. In numerischen Experimenten wird gezeigt, dass die neuen Algorithmen für die meisten Funktionen weniger Funktionsauswertungen als existierende Methoden benötigen und dennoch die gleiche Genauigkeit erzielen. Zudem wird eine Spektralmethode zur Lösung partieller Differenzialgleichungen entwickelt, die mit funktionalen Niedrigrangapproximationen kombiniert werden könnte.

Anschliessend befasst sich diese Arbeit mit dem optimalen Transport mehrerer Randverteilungen. Das zugehörige mathematische Problem kann durch entropische Regularisierung in ein Tensorskalierungsproblem umgewandelt werden, das wiederum mit dem Sinkhorn-Algorithmus gelöst werden kann. In der Literatur existieren Ansätze, diesen Algorithms entweder durch das Ausnutzen eines probabilistischen grafischen Modells oder durch eine Niedrigrangapproximation zu beschleunigen. Diese beiden Ansätze werden in dieser Arbeit kombiniert, um den Sinkhorn-Algorithmus noch stärker zu beschleunigen.

Zuletzt wird das Problem der Berechnung von Selbstdiffusionsmatrizen stochastischer Prozesse mit markiertem Teilchen auf Gittern untersucht. In dieser Arbeit wird zur Lösung dieses Problems ein neuartiger Ansatz basierend auf der Lösung eines hochdimensionalen Minimierungsproblems präsentiert. Im Vergleich zu klassischen Algorithmen, die die langfristige gemittelte Abweichung eines stochastischen Prozesses mit einer Monte-Carlo-Simulation bestimmen, führt der neuartige Algorithmus in numerischen Experimenten zu deutlich weniger statistischem Rauschen.

Stichwörter: Niedrigrangapproximation, hochdimensionale Probleme, Tensormethoden, Fluch der Dimensionalität, Tschebyscheff-Interpolation, funktionale Niedrigrangapproximation, optimaler Transport mehrerer Randverteilungen, Selbstdiffusionsmatrix, Spektralmethode

Contents

Acknowledgements **i**

Abstract **iii**

Zusammenfassung **v**

1 Introduction **1**

2 Preliminaries **7**
- 2.1 Low-rank approximation . . . 7
  - 2.1.1 Low-rank approximation of matrices . . . 7
  - 2.1.2 Low-rank approximation of tensors . . . 11
  - 2.1.3 Low-rank tensor formats with additional structure . . . 12
  - 2.1.4 Approximation algorithms . . . 13
- 2.2 Functional low-rank approximation . . . 15
- 2.3 Chebyshev interpolation . . . 17
  - 2.3.1 Chebyshev interpolation for univariate functions. . . . 17
  - 2.3.2 Tensorized Chebyshev interpolation for multivariate functions . . . 18
  - 2.3.3 Coefficient approximation . . . 20
  - 2.3.4 Connection to functional low-rank approximations . . . 22
  - 2.3.5 Proof of concept . . . 22

3 Functional Tucker approximation **25**
- 3.1 Existing algorithm: Chebfun3 . . . 26
  - 3.1.1 Phase 1: Block term decomposition . . . 26
  - 3.1.2 Phase 2: Refinement . . . 27
  - 3.1.3 Phase 3: Compression . . . 27
  - 3.1.4 Disadvantages . . . 28
- 3.2 Novel algorithm: Chebfun3F . . . 29
  - 3.2.1 Phase 1: Fiber indices and factor matrices . . . 29
  - 3.2.2 Phase 2: Refinement of the factors . . . 31
  - 3.2.3 Phase 3: Reconstruction of the core tensor . . . 31
  - 3.2.4 Chebfun3F algorithm . . . 32
  - 3.2.5 Existence of a quasi-optimal Chebfun3F approximation . . . 34

3.2.6 Comparison of the theoretical cost 35
3.3 Numerical results 36
3.3.1 Chebfun3 vs. Chebfun3F 36
3.3.2 Comparison to sparse grids 39
3.4 Coefficient decay analysis 41
3.4.1 Observations in a two-dimensional setting 42
3.4.2 Potential solutions in the two-dimensional setting 43
3.4.3 Incorporation of Chebfun3F in Chebfun 44
3.5 Additional insights 45
3.5.1 Approximations in Lebesgue spaces 45
3.5.2 Further compression 46

4 Extended functional tensor train approximation 51
4.1 Extended functional tensor train format 52
4.2 Approximation algorithm 54
4.3 Numerical experiments 56
4.3.1 Comparison to a direct TT approximation 57
4.3.2 Comparison to the FTT approximation algorithm 59
4.A Appendix 63
4.A.1 Benchmark functions 63
4.A.2 Genz functions 65
4.A.3 Parametric PDE problem 65

5 Multi-marginal optimal transport 67
5.1 Multi-marginal optimal transport and the Sinkhorn algorithm 68
5.1.1 Mathematical setting 69
5.1.2 Multi-marginal Sinkhorn algorithm 69
5.2 Impact of approximating the Gibbs kernel 72
5.3 Tensor networks and graphical models 76
5.4 Low-rank approximations in tensor networks 79
5.5 Numerical experiments 80
5.5.1 Proof of concept 80
5.5.2 Application: Color transfer from color barycenters 83
5.5.3 A tensor network with circles 86

6 Self-diffusion matrix 91
6.1 Infinite-dimensional definition 92
6.1.1 Definition as optimization problem 92
6.1.2 Definition as long time mean square deviation 93
6.2 Finite-dimensional approximation 94
6.2.1 Discretized minimization problems 94
6.2.2 Combined minimization problem 95
6.2.3 Estimation of long-time mean square deviation 96

6.3 Low-rank solutions for the optimization problem 97
  6.3.1 Fast and stable evaluation 98
  6.3.2 Successive minimization 99
  6.3.3 Alternating least squares 100
  6.3.4 Monte Carlo methods 102
  6.3.5 Limitations of the approach 102
6.4 Numerical Experiments 104
  6.4.1 Solving the optimization problem 104
  6.4.2 Estimation of long-time mean square deviation 105
  6.4.3 Comparison of algorithms 107
6.5 Application: Cross-diffusion system 109
  6.5.1 Hydrodynamic limit of a lattice-based stochastic hopping model . . 109
  6.5.2 Deterministic resolution of a simplified cross-diffusion system . . . 111
  6.5.3 Numerical Experiment 114

7 Fast global spectral method **117**
7.1 Problem setting 119
  7.1.1 Structure of a linear differential operator 119
  7.1.2 Approximation format 119
7.2 Operator discretization 119
  7.2.1 One-dimensional differential operators 120
  7.2.2 Three-dimensional differential operators 122
  7.2.3 Generalization to non-constant coefficients 122
7.3 A spectral method for three-dimensional linear PDEs 125
  7.3.1 Boundary condition discretization 126
  7.3.2 Incorporating the boundary conditions 127
  7.3.3 Solving tensor-valued linear systems 129
7.4 Numerical results 130
  7.4.1 Runtime comparison 130
  7.4.2 Stationary problems 131
  7.4.3 Time-dependent problems 137
  7.4.4 Eigenvalue problems 138

8 Conclusions and outlook **141**

Bibliography **145**

Curriculum Vitae **167**

1 Introduction

This thesis is concerned with the solution of three different high-dimensional problems: the approximation of multivariate functions using tensorized Chebyshev interpolation, the solution of entropically regularized multi-marginal optimal transport problems, and the computation of self-diffusion matrices via solving a high-dimensional optimization problem.

At a first glace, the problems studied in this thesis might seem vastly different, but at heart all of them involve a tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ of order d, where d denotes the dimension of the problem. Since the size of $\mathcal{T}$ increases exponentially with respect to d, we can not store the tensor $\mathcal{T}$ any more for large values of d. To potentially mitigate this so-called curse of dimensionality [33], we need to identify and exploit underlying structures of $\mathcal{T}$. For instance, when $\mathcal{T}$ is sparse, we only need to store the non-zero entries. When $\mathcal{T}$ is a rank-1 tensor, i.e.

$$\mathcal{T}_{i_1,i_2,\ldots,i_d} = u^{(1)}_{i_1} u^{(2)}_{i_2} \cdots u^{(d)}_{i_d}, \qquad \text{for } i_\ell = 1,\ldots,n_\ell,\ \ell = 1,\ldots,d, \tag{1.1}$$

for some vectors $u^{(\ell)} \in \mathbb{R}^{n_\ell}$, $\ell = 1,\ldots,d$, we only need to store these vectors. In applications, the tensor $\mathcal{T}$ rarely has rank-1 structure, but often $\mathcal{T}$ can be well approximated by the sum of r rank-1 tensors, i.e. by a tensor of rank at most r [190]. The resulting so-called low-rank tensor approximations [145, 198] have been applied to solve a plethora of high-dimensional problems in various fields including but not limited to model order reduction [243], uncertainty quantification [203], sensitivity analysis [201], machine learning [179, 291], signal processing [72], the solution of partial differential [22, 188] and time-dependent differential equations [63, 196], data science [302], optimization [73, 274], quantum physics [256, 331], plasma physics [109] and quantum chemistry [312]. In certain applications, it is beneficial to impose additional structure to the sum of separable functions, leading to the so-called Tucker format [327], the tensor train format [249] or general tensor networks [247]. We would like to point out that it is even possible to apply low-rank tensor approximations to low-dimensional problems by reshaping matrices and vectors into tensors [187, 248].

Functional low-rank approximation

The basic ideas of low-rank approximations can be generalized from tensors in finite dimensional tensor product spaces to multivariate functions defined on tensor product domains [154]. Instead of sums of rank-1 tensors, we are now interested in sums of separable functions. A function $f : [-1,1]^d \to \mathbb{R}$ is called separable if it can be written in terms of univariate functions $g_\ell : [-1,1] \to \mathbb{R}$ for $\ell = 1, \dots, d$ as

$$f(x_1, x_2, \dots, x_d) = g_1(x_1)g_2(x_2)\cdots g_d(x_d), \qquad \text{for } x_\ell \in [-1,1],\ \ell = 1, \dots, d. \tag{1.2}$$

For the sake of simplicity, we only consider the domain $[-1,1]^d$ throughout this thesis. Generalizations to arbitrary tensor product domains are possible via appropriate domain transformations. Further extensions to spheres are also viable [43, 321]. It is again possible to prescribe additional structure to the sum of separable functions to obtain approximations in the so-called functional Tucker [100, 163] or functional tensor train format [41, 141]. The structure of the functional low-rank approximation can then be exploited in quite a few applications, including the solution of time-dependent partial differential equations (PDEs) [91], uncertainty quantification [202], sensitivity analysis [25], optimal control [142] and quantum dynamic simulations [298].

It is important to emphasize that not every function admits a good approximation in low-rank formats with a moderate amount of terms, especially for larger d. Generally speaking, smoothness and a restricted (e.g., nearest neighbor) interaction between variables are helpful. Upper bounds for the number of terms needed to attain a certain accuracy are derived in [41, 146] for functions in Sobolev spaces. The required number of terms can change significantly when variables are transformed [324] or the order of variables is permuted [92]. For functions in periodic and mixed Sobolev spaces approximation rates can be found in [286] and [148], respectively. Much faster rates can be obtained for functions with special structures such as compositional functions [21] and quantities associated with certain parametric PDEs [20]. A more abstract analysis of functional low-rank approximation can be found in [4, 5, 6].

In practice, we obtain a fully discrete approximation of f via discretizing each univariate function in the approximation by, e.g. a truncated series expansion, which incurs an additional truncation error [141]. The resulting approximations are intricately related to approximations of f using a truncated expansion in terms of tensorized basis functions of the form

$$f(x_1, x_2, \dots, x_d) \approx \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} \mathcal{A}_{i_1,i_2,\dots,i_d} b_{i_1}^{(1)}(x_1) b_{i_2}^{(2)}(x_2) \cdots b_{i_d}^{(d)}(x_d), \tag{1.3}$$

with coefficient tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_d}$ of order d and univariate basis functions $b_i^{(\ell)} : [-1, 1] \to \mathbb{R}$ for $i = 1, \ldots, n_\ell,\ \ell = 1, \ldots, d$. Any low-rank approximation of $\mathcal{A}$ can be seen as functional low-rank approximation of f. For instance, the Chebfun package [104] for computing numerically on the level of functions [260] approximates bi- and trivariate functions [163, 320] by computing low-rank approximations of the coefficient tensor obtained via multivariate interpolation [209] with tensorized Chebyshev polynomial basis functions [223]. The univariate functions in the resulting functional low-rank approximation are internally represented in terms of univariate Chebyshev interpolants [38, 165]. Once the functional low-rank approximation is obtained, operations such as computing derivatives are performed by manipulating the Chebyshev coefficients of the univariate interpolants [27]. Note that the interpolation approach in Chebfun requires that we have access to point evaluation on a grid. When only point evaluations at unstructured points are available, we can use supervised learning to obtain functional low-rank approximations [140].

Instead of computing low-rank approximations of $\mathcal{A}$, we can also try to choose the basis functions b_i^ℓ in (1.3) such that most of the information about the function is stored in a low number of entries of $\mathcal{A}$, i.e. we can approximate $\mathcal{A}$ by a sparse tensor. This is the idea of sparse girds [295, 345]. Especially for functions with bounded mixed derivatives, the sparse grid approach offers the potential to mitigate the curse of dimensionality [50].

We briefly want to point out that there exist many other techniques that offer the potential to mitigate or even overcome the curse of dimensionality when approximating a high-dimensional function $f : [-1, 1]^d \to \mathbb{R}$. For instance, we could decompose a multivariate function using the ANOVA-decomposition [297], which represents f as truncated sum of functions depending on small subsets of variables [296], or using arithmetic circuit tensor networks [254], which describe the multivariate function as the interaction of univariate functions in terms of a tensor network. Approximations of f obtained using neural networks can implicitly exploit certain structures of f to overcome the curse of dimensionality [19, 262]. Dimensionality reductions techniques such as active subspaces [76], inverse regression [212] and autoencoders [167] can be used to identify whether f actually depends on all variables or whether it is mostly determined by a low-dimensional subspace of $[-1, 1]^d$. In the latter case, we can replace the high-dimensional f by a low-dimensional surrogate defined directly on the subspace. However, none of these alternative approaches yields an approximations in terms of sums of univariate functions, which would be highly beneficial for further numerical computations [39, 40].

Multi-marginal optimal transport

Classical optimal transport minimizes the transport cost between $d = 2$ probability measures [34, 333]. For discrete measures, this problem can be expressed as the minimization of $\langle C, P \rangle$ for a nonnegative cost matrix C. The so-called transport plan P is a nonnegative matrix that has to satisfy marginal constraints, i.e. the column and row sums of P are prescribed by discrete measures. The pioneering work of Cuturi [82] established a relation between entropy regularized optimal transport and matrix scaling of $\exp(-C/\eta)$, where exp denotes the elementwise exponential and $\eta > 0$ denotes the regularization parameter. Scaling the rows and columns of $\exp(-C/\eta)$ such that marginal constraints are satisfied can be achieved numerically using the Sinkhorn algorithm [294], whose convergence speed can be accelerated using greedy coordinate descent [9, 216], overrelaxation [314] or accelerated gradient descent [106]. This matrix scaling approach allows one to solve much larger optimal transport problems compared to previous attempts based on solving the original linear program, which in turn has impacted various fields including image processing [121, 267], data science [258], engineering [235] and machine learning [134, 199].

The classical optimal transport problem can be generalized to a multi-marginal setting, in which the transport cost between $d \geq 3$ measures is minimized [253]. Such multi-marginal problems arise in the areas of density functional theory [97], generalized incompressible flow [36], neural networks [57], signal processing [110] and Wasserstein barycenters [59]. For discrete measures, the problem can be expressed as finding the nonnegative transport plan tensor $\mathcal{P}$ of order d, which minimizes $\langle \mathcal{C}, \mathcal{P} \rangle$ subject to marginal constraints, where $\mathcal{C}$ denotes a given nonnegative cost tensor of order d. In analogy to the matrix case, after adding entropic regularization, the problem can equivalently be transformed into a tensor scaling problem for the tensor $\exp(-\mathcal{C}/\eta)$ [37]. The Sinkhorn algorithm can be generalized to solve this multi-marginal problem [58]; acceleration techniques via greedy coordinate descent are described in [127, 217].

The multi-marginal Sinkhorn algorithm crucially relies on the repeated evaluation of marginals of the rescaled tensor $\exp(-\mathcal{C}/\eta)$. The cost of computing such a marginal increases exponentially with respect to d. There exist two different approaches in literature to mitigate this curse of dimensionality by exploiting additional structure of $\exp(-\mathcal{C}/\eta)$. The first line of research is to exploit that, in many applications, the structure of $\exp(-\mathcal{C}/\eta)$ allows to specify the transport plan in terms of a graphical model. When this graphical model does not contain circles, marginals can be computed efficiently using the belief propagation algorithm [118, 150] and Fourier-based fast summation [17]. In particular, this includes tree structured cost tensors [32, 151]. When the model contains circles, the junction tree algorithm [173] can be used to evaluate the marginals, but it might still incur a large computational cost. A second approach to possibly attain a complexity reduction is to replace $\exp(-\mathcal{C}/\eta)$ by a low-rank approximation, whose marginals can be evaluated efficiently [8]. For the classical case $d = 2$, Altschuler et al. [10] analyze the impact of the approximation error on the solution returned by the Sinkhorn algorithm. For the case $d \geq 3$, asymptotic complexity bounds are derived in [7] for the specific case that $\mathcal{C}$ has low tensor rank [190] and is given explicitly in factored form. Let us point out that low-rank approximations of $\exp(-\mathcal{C}/\eta)$ should not be confused with low-rank approximations of the desired transport plan, as proposed in [283, 284]. Further, we would like to point out that it is also possible to exploit the structure of $\mathcal{C}$ [348] and sparsity of the solution [129, 130] to efficiently compute solutions of the unregularized multi-marginal optimal transport problem.

Self-diffusion matrix

The self-diffusion matrix of a tagged particle process on a grid [200] identifies the hydrodynamic limit of multi-species symmetric exclusion processes [266]. It is traditionally computed by approximating long-time averages of the mean-square displacement of the tagged particle [42, 89, 193, 266]. In a first step, the computational domain is truncated to a finite-size supercell with periodic boundary conditions. Then, a large number of realisations of the trajectories of the tagged particle are sampled in order to approximate the mean-square displacement by an empirical average computed with a standard Monte-Carlo approach [158, 227]. Moreover, the value of a finite final time has to be chosen beforehand to approximate the long-time limit. In [210], it is shown that the error linked to the truncation of the computational domain decays exponentially with the size of the supercell. In contrast, the statistical error of the approximation linked to the use of a finite number of random samples of the trajectories of the tagged particle decays as the inverse of the square root of the number of samples. As a consequence, the main source of error in the practical computation of approximations of the self-diffusion matrix is due to statistical noise; see e.g. [95, 120, 159, 279, 315, 317].

The self-diffusion matrix can equivalently be defined via the solution of a deterministic high-dimensional optimization problem [42, 210, 266] of the form

$$\min_{\mathcal{T}\in\mathbb{R}^{2\times 2\times\cdots\times 2}} f(\mathcal{T}),$$

where $f : \mathbb{R}^{2\times 2\times\cdots\times 2} \to \mathbb{R}$ denotes the objective function and the order d of $\mathcal{T}$ is equal to the number of grid points in the finite-size supercell. Solving this minimization problem potentially yields the self-diffusion matrix without incurring any statistical noise. In order to solve this minimization problem numerically, we need to reduce the number of degrees of freedom. This can be achieved by assuming that $\mathcal{T}$ can be well approximated by a sum of rank-1 tensors. We can then iteratively optimize each of the terms in the low-rank representation of $\mathcal{T}$ using a classical alternating linear scheme [171, 274]. Alternatively, we could use Riemannian optimization [44, 205, 264] to solve the optimization problem directly on the manifold of low-rank tensors directly.

Organisation of the thesis

Chapter 2. We recall the basic concepts of low-rank tensor approximations, low-rank functions and tensorized Chebyshev interpolation and establish the notation for the remainder of this thesis.

Chapters 3 and 4 (based on [100] and [306]). We develop two novel algorithms for computing functional low-rank approximations by combining tensorized Chebyshev interpolation with low-rank approximations of the coefficient tensor in the Tucker and extended tensor train format. Compared to existing algorithms our novel approaches require fewer function evaluations to achieve the same approximation accuracy.

Chapter 5 (based on [305]). We propose to combine graphical models and low-rank approximations to further accelerate the computation of marginals for the Sinkhorn algorithm. Our numerical experiments demonstrate that this combination offers the potential to solve of entropically regularized optimal transport problems even faster.

Chapter 6 (based on [85] and [84]). We develop a novel approach to compute self-diffusion matrices numerically by solving the high-dimensional minimization problem using an alternating linear scheme instead of sampling long-time averages of the mean-square displacement.

Chapter 7 (based on [304]). We develop a global spectral method to solve three-dimensional PDEs using discretizations of the form (1.3).

Chapter 8. We conclude the thesis and provide an outlook on further research directions. In particular, we discuss the potential to combine the spectral method presented in Chapter 7 with the low-rank approximations of the coefficient tensor in Chapters 3 and 4.

2 Preliminaries

In this chapter, we recall the notation and basic results for low-rank approximations of matrices, tensors and functions as well as the fundamental concepts for tensorized Chebyshev interpolation.

In Section 2.1, we introduce low-rank matrices and tensors. We establish a notation for low-rank approximations of matrices and tensors in various formats following the book by Golub and Van Loan [137] and the article by Kolda and Bader [198].

In Section 2.2, we generalize the concept of low-rank approximations from tensors to multivariate functions and introduce the corresponding functional low-rank approximation formats.

In Section 2.3, we recall the fundamentals of Chebyshev interpolation for univariate functions from the book by Mason and Handscomb [224] and the book by Trefethen [322]. We extend the concept to multivariate functions using tensorization. The resulting interpolants are computed based on a tensor containing the evaluation of the function on a tensor product grid. We study how approximating this tensor affects the interpolation error and how such approximation are related to functional low-rank approximations.

2.1 Low-rank approximation

2.1.1 Low-rank approximation of matrices

A matrix $A \in \mathbb{R}^{m \times n}$ is said to be rank-1, when it can be written as $M = u \circ v$ for vectors $u \in \mathbb{R}^m,\ v \in \mathbb{R}^n$, where the outer product $\circ$ is defined as

$$(u \circ v)_{i,j} = u_i v_j, \qquad \text{for } i = 1, \dots, m,\ j = 1, \dots, n.$$

In general, the rank r of a matrix $A \in \mathbb{R}^{m\times n}$ is defined as the minimum $r \in \mathbb{N}$ such that A can be written as a sum of r rank-1 matrices. In the following, we focus on the approximation of given matrix $A \in \mathbb{R}^{n\times m}$ by a matrix $B \in \mathbb{R}^{n\times m}$ subject to the constraint that the rank of B is at most r for a fixed $r \in \mathbb{N}$. The matrix B is called low-rank approximation of A.

Since $\operatorname{rank}(B) \leq r$, there exist matrices $U \in \mathbb{R}^{m\times r},\ V \in \mathbb{R}^{n\times r}$ such that $B = UV^T$. Assume that $r \ll \min(n, m)$. We observe that storing the matrices U, V of the low-rank approximation B requires only $\mathcal{O}(r(m + n))$ storage, whereas the full matrix A requires $\mathcal{O}(mn)$ storage. By replacing the matrix A by the low-rank approximation B in subsequent computations, we can potentially accelerate these computations by exploiting the structure of B. For instance, matrix vector products involving B can be performed in $\mathcal{O}(r(m + n))$ operations when B is given in terms of the matrices U, V, whereas matrix vector productions involving A require $\mathcal{O}(mn)$ operations. Note that this introduces an error depending on how well A is approximated by B.

Let $\|\cdot\|_2$ denote the spectral norm and let $\|\cdot\|_F$ denote the Frobenius norm. The Eckhart-Young-Mirsky theorem [231] states that an optimal rank-r matrix B in the sense that $\|A - B\|_2$ or $\|A - B\|_F$ is minimized can be computed by truncating the singular value decomposition of A. Finding this so-called best approximation with respect to the maximum norm $\|\cdot\|_\infty$ or the entrywise ℓ_1-norm $\|\cdot\|_1$ is still subject to research [299, 343]. Note that we do not necessarily need to compute the best rank-r approximation to accelerate subsequent computations. It suffices to find any matrices $U \in \mathbb{R}^{m\times r},\ V \in \mathbb{R}^{n\times r}$ such that $A \approx UV^T$. In the following, we present how to compute such low-rank approximations of A efficiently. The presented approximations can be computed faster than the truncated singular value decomposition and can potentially be obtained without evaluating every single entry of A. For a detailed overview of low-rank approximation algorithms for matrices, we refer to the review [194]. Further, we would like to point out that it is even possible to compute low-rank approximations when only selected entries of A can be accessed [55, 350].

We now introduce so-called cross approximations [138]. These are rank-r approximations of the form

$$A \approx A(:, J)A(I, J)^{-1}A(I, :), \tag{2.1}$$

where $I \subset \{1, \dots, m\},\ J \subset \{1, \dots, n\}$ denote index sets of cardinality r chosen such that $A(I, J)$ is invertible. This approximation format is visualized in Figure 2.1. Throughout this thesis, we use adaptive cross approximation (ACA) [30] to determine suitable index sets. Initially, ACA determines the index tuple (i, j) of the entry with maximal absolute value of A and adds i to the initially empty index set I and j to J respectively. This greedy approach is then repeated iteratively by determining the largest absolute value of the residual of the current cross approximation. The algorithm is stopped once the

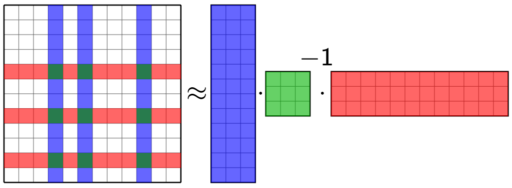


Figure 2.1 – Visualization of a cross approximation (2.1). The columns $A(:,J)$ are marked in blue. The rows $A(I,:)$ are marked in red.

largest absolute value of the residual falls below a prescribed tolerance. This adaptively determines the suitable rank-r. We formalize ACA in Algorithm 1. Note that ACA corresponds to greedily maximizing the volume of $A(I,J)$ [138] and comes with theoretical guarantees [80]. An overview of alternative algorithms to determine the index sets is given in [78].

Algorithm 1 Adaptive cross approximation

```
1: Input: matrix $A \in \mathbb{R}^{m\times n}$, tolerance $\varepsilon$
2: Output: index sets $I$ and $J$ s.t. $A \approx A(:,J)A(I,J)^{-1}A(I,:)$
3: $I = []$, $J = []$
4: while $\max|A| \geq \varepsilon$
5:     $(i,j) = \arg\max_{(i,j)} |A(i,j)|$
6:     $I = [I,i]$, $J = [J,j]$
7:     $A = A - A(:,j)A(i,:)/A(i,j)$
```

Remark 2.1. *In Algorithm 1, we present ACA with full pivoting, i.e. we determine the next index tuple based on the largest absolute value in line 4. This requires the evaluation of every single entry of A. In practice, we could use any entry with a large absolute value to determine the next index tuple. A commonly used heuristic to determine entries with large absolute value is partial pivoting [29]. An initially random index tuple* (i,j) *is updated repeatedly by alternatingly setting* $i = \arg\max_i |A(i,j)|$ *and* $j = \arg\max_j |A(i,j)|$. *This process only requires the evaluation of several rows and columns instead of the whole matrix.*

Cross approximations can be generalized to so-called CUR-decompositions [45] by allowing for an arbitrary matrix $\tilde{U} \in \mathbb{R}^{r\times r}$ instead of the inverse of $A(I,J)$. This leads to an approximation of the form

$$A \approx A(:,J)\tilde{U}A(I,:). \tag{2.2}$$

Given the matrix $C = A(:,J)$ containing columns of A and the matrix $R = A(I,:)$ containing rows of A, we can project the columns of A onto the span of C and the rows of A onto the span of R using orthogonal projections. This yields $A \approx CC^TAR^TR$, which corresponds to the CUR decomposition (2.2) with $\tilde{U} = C^TAR^T$. Note that computing C^TAR^T requires the evaluation of the full matrix A.

In the following, we describe how we can replace the orthogonal projections by oblique projections to obtain a low-rank approximation of A given C, R without fully evaluating A. Let Q_C, Q_R denote the orthogonal matrices in the economic QR decomposition of C, R respectively. If we use oblique projections [300] based on index sets $\tilde{I} \subset \{1, \ldots, m\}$, $\tilde{J} \subset \{1, \ldots, n\}$ of cardinality r chosen such that $Q_C(\tilde{I}, :)$ and $Q_R(\tilde{J}, :)$ are invertible, we obtain

$$\begin{aligned} A &\approx Q_C(\Phi_{\tilde{I}}^T Q_C)^{-1}\Phi_{\tilde{I}}^T A(Q_R(\Phi_{\tilde{J}}^T Q_R)^{-1}\Phi_{\tilde{J}}^T)^T \\ &= Q_C(Q_C(\tilde{I}, :)^{-1} A(\tilde{I}, \tilde{J})(Q_R(\tilde{J}, :)^{-T} Q_R, \end{aligned} \tag{2.3}$$

where $\Phi_{\tilde{I}}^T A = A(\tilde{I}, :)$. Note that (2.3) is a rank-r approximation of A, which given C, R can be computed by only evaluating r^2 entries of A. The following lemma summarizes the statement of [66, Lemma 7.3], which plays a critical role in guiding the choice of the indices $\tilde{I}, \tilde{J}$.

Lemma 2.1. *Let $A \in \mathbb{R}^{m \times n}$, $m \geq n$, have orthonormal columns. Consider an index set $I \subset \{1, \ldots, m\}$ of cardinality n such that $\Phi_I^T A$ is invertible. Then the oblique projection $A(\Phi_I^T A)^{-1}\Phi_I^T$ satisfies*

$$\|x - A(\Phi_I^T A)^{-1}\Phi_I^T x\|_2 \leq \|(\Phi_I^T A)^{-1}\|_2 \cdot \|(I - AA^T)x\|_2, \quad \forall x \in \mathbb{R}^n.$$

Lemma 2.1 exhibits the critical role played by the quantity $\|(\Phi_I^T Q_C)^{-1}\|_2 \geq 1$ for oblique projections. Suitable index sets $\tilde{I}, \tilde{J}$ given Q_C, Q_R can be computed using the discrete empirical interpolation method (DEIM) [65, 66], presented in Algorithm 2. In practice, the computed index sets usually yield good approximations as $\|(\Phi_I^T Q_C)^{-1}\|_2$ tends to be small. Note that we could in principle apply oblique projections using C, R directly instead of Q_C, Q_R, but $(\Phi_I^T C)^{-1}$ is potentially ill-conditioned.

Algorithm 2 Discrete empirical interpolation method

```
Input: orthonormal matrix $M \in \mathbb{R}^{m \times n}$
Output: index set $I$
$I = [\mathsf{argmax}\ |A(:, 1)|]$
for  $k = 2, \ldots, n$
    $c = A(I, 1:k-1) \setminus A(I, k)$
    $r = A(:, k) - A(:, 1:k-1)c$
    $I = [I, \mathsf{argmax}\ |r|]$
```

Remark 2.2. *Note that not all matrices can be well approximated by low-rank matrices. It might, however, be possible to approximate their off-diagonal blocks by low-rank matrices. Storing these approximations instead of the full blocks reduces the required storage. The resulting so-called hierarchical matrices [31, 155] are, in particular, useful for the approximation of invertible matrices, since solutions of linear systems involving hierarchical matrices can be computed efficiently.*

2.1.2 Low-rank approximation of tensors

Let $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ denote a rank-1 tensor represented by vectors $u^{(\ell)} \in \mathbb{R}^{n_\ell}$, $\ell = 1, \ldots, n$ as in (1.1). Using the multiple outer product, we can write this tensor as $\mathcal{T} = u^{(1)} \circ u^{(2)} \circ \cdots \circ u^{(d)}$. In general, the tensor-rank r of the tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is defined as the minimum $r \in \mathbb{N}$ such that $\mathcal{T}$ can be written as sum of r rank-1 tensors [168]. A tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_d}$ with rank at most r is said to be represented in CANDECOMP/PARAFAC (CP)-format [190] if it is given in terms of vectors $u^{(\ell,k)} \in \mathbb{R}^{n_\ell}$ for $\ell = 1, \ldots, d,\ k = 1, \ldots, r$ as

$$\mathcal{T} = \sum_{k=1}^{r} u^{(1,k)} \circ u^{(2,k)} \circ \cdots \circ u^{(d,k)}. \tag{2.4}$$

Note that storing these vectors only requires $\mathcal{O}(drn)$ storage, whereas storing the entries of $\mathcal{T}$ directly requires $\mathcal{O}(n^d)$ storage, where we assume that $n_1 = n_2 = \cdots = n_d = n$.

In this work, we are primarily interested in using low-rank approximations of tensors to accelerate computations. As in the matrix case discussed in Section 2.1.1, we do not need to solve the hard problem [252, 292] of finding the best rank-r approximation with respect to a tensor norm such as the Frobenius norm $\|\cdot\|_F$, the entrywise ℓ_1-norm $\|\cdot\|_1$ or the maximum norm $\|\cdot\|_\infty$. Instead any reasonably accurate low-rank approximation suffices.

The alternating least squares (ALS) algorithm [60, 161] is a commonly used alternating linear scheme [39, 145, 250, 274] to compute approximations in the CP-format (2.4). Let $\tilde{\ell} \in \{1, \ldots, d\}$. We observe that the problem of minimizing the approximation error

$$\|\mathcal{T} - \sum_{k=1}^{r} u^{(1,k)} \circ u^{(2,k)} \circ \cdots \circ u^{(d,k)}\|_F$$

is a linear least squares problem when we fix $u^{(\ell,k)}$ for $k = 1, \ldots, n,\ \ell = 1, \ldots, d,\ \ell \neq \tilde{\ell}$ and only optimize with respect to $u^{(\tilde{\ell},1)}, \ldots, u^{(\tilde{\ell},r)}$. This allows us to improve a given approximation by updating $u^{(\tilde{\ell},1)}, \ldots, u^{(\tilde{\ell},r)}$ based on the solution of this least squares problem. Performing such updates repeatedly for alternating values of $\tilde{\ell}$ leads to the ALS algorithm. Note that applying ALS to a randomly initialized approximation might not converge [198]. Still, this approach yields useful low-rank approximations in practice.

Remark 2.3. *The ALS algorithm requires a priori knowledge of a suitable rank-r for the CP approximation* (2.4)*. Note that the sum of two tensors in CP-format can again be represented in CP-format. This allows us to refine a given approximation by adding an approximation of the residual of the current approximation in CP-format. Performing this process successively leads to a rank-adaptive approximation algorithm [11, 145, 242].*

Remark 2.4. *We would like to point out that additional constraints can be imposed on the vectors* $u^{(\ell,k)}$ *in* (2.4)*. For smoothness, nonnegativity and sparseness constraints we refer to [131]. For imposed linear dependencies we refer to the overview [119].*

2.1.3 Low-rank tensor formats with additional structure

Prescribing additional structure to the sum (2.4) leads to different approximation formats. In the following, we introduce two commonly used formats.

Let $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$. A vector $v \in \mathbb{R}^{n_\ell}$ is called mode-ℓ fiber of $\mathcal{T}$ if there exist indices $i_k \in \{1, \ldots, n_d\}$ for $k = 1, \ldots, d,\ k \neq \ell$ such that $\mathcal{T}_{i_1,\ldots,i_\ell,\ldots,i_d} = v_{i_\ell}$ for all $i_\ell = 1, \ldots, n_\ell$. We denote by $\mathcal{T}^{\{\ell\}} \in \mathbb{R}^{n_\ell \times (n_1 \cdots n_{\ell-1} \cdot n_{\ell+1} \cdots n_d)}$ the matrix containing all mode-ℓ fibers of $\mathcal{T}$ in its columns, i.e.

$$\mathcal{T}^{\{\ell\}}_{i_\ell, i_{\neq \ell}} = \mathcal{T}_{i_1,i_2,\ldots,i_d}, \qquad \text{where } i_{\neq\ell} = 1 + \sum_{\substack{j=1 \\ j \neq \ell}}^{d} \Big((i_j - 1) \prod_{\substack{k=1 \\ k \neq \ell}}^{j-1} n_k \Big),$$

for all $i_\ell = 1, \ldots, n_\ell,\ \ell = 1, \ldots, d$. This matrix is known as mode-ℓ matricization of $\mathcal{T}$. The multilinear rank $(r_1, r_2, \ldots, r_d) \in \mathbb{N}^d$ of $\mathcal{T}$ is defined based on these matricizations as $r_\ell = \operatorname{rank}(\mathcal{T}^{\{\ell\}})$ for $\ell = 1, \ldots, d$.

A tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ with multilinear rank at most $(r_1, r_2, \ldots, r_d)$ is said to be represented in Tucker format [327] if it is given in terms of a so-called core tensor $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}$ and so-called factor matrices $U^{(\ell)} \in \mathbb{R}^{n_\ell \times r_\ell}$ for $\ell = 1, \ldots, d$ as

$$\mathcal{T}_{i_1,i_2,\cdots,i_d} = \sum_{k_1=1}^{r_1} \sum_{k_2=1}^{r_2} \cdots \sum_{k_d=1}^{r_d} \mathcal{C}_{k_1,k_2,\ldots,k_d} U^{(1)}_{i_1,k_1} U^{(2)}_{i_2,k_2} \cdots U^{(d)}_{i_d,k_d},$$

for $i_\ell = 1, \ldots, n_\ell,\ \ell = 1, \ldots, d$. This can be written as

$$\mathcal{T} = \mathcal{C} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 \cdots \times_d U^{(d)}, \tag{2.5}$$

where $\times_\ell$ denotes the mode-ℓ multiplication. For a tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ and a matrix $M \in \mathbb{R}^{m \times n_\ell}$ it is defined as the multiplication of every mode-ℓ fiber of $\mathcal{T}$ with M, i.e.

$$(\mathcal{T} \times_\ell M)_{i_1,\ldots,i_d} = \sum_{j_\ell=0}^{n_\ell} \mathcal{T}_{i_1,\ldots,i_{\ell-1},j_\ell,i_{\ell+1},\cdots,i_d} M_{i_\ell,j_\ell}, \qquad \text{for } i_\ell = 1, \ldots, n_\ell,\ \ell = 1, \ldots, d.$$

This is related to the matrizisation in the following way $(\mathcal{T} \times_\ell M)^{\{\ell\}} = M\mathcal{T}^{\{\ell\}}$. Storing a tensor in Tucker format requires $\mathcal{O}(r^d + dnr)$ storage under the assumption that $n_1 = n_2 = \cdots = n_d = n$ and $r_1 = r_2 = \cdots = r_d = r$.

Remark 2.5. *A special type of Tucker decompositions are so-called tensor CUR decompositions [53], which generalize matrix CUR decompositions* (2.2) *to tensors by requiring that the columns of the matrices $\mathcal{U}^{(\ell)}$ contain mode-ℓ fibers of the tensor $\mathcal{T}$. It is also possible to impose nonnegativity constraints on either the whole Tucker approximation [180, 309] or the individual factor matrices [191, 349] when $\mathcal{T}$ is nonnegative.*

The second low-rank approximation format of interest are so-called tensor train (TT) decompositions [249]. These are also known as matrix product states [256, 337] in the Physics community.

Let $\mathcal{T}^{<\ell>}$ denote the tensor obtained by reshaping $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ into $\mathbb{R}^{(n_1 \cdot n_2 \cdots n_\ell) \times (n_{\ell+1} \cdots n_d)}$ for $\ell = 1, \ldots, d-1$. The TT ranks $(R_1, \ldots, R_{d-1}) \in \mathbb{N}^{d-1}$ of $\mathcal{T}$ are defined as $R_\ell = \operatorname{rank}(\mathcal{T}^{<\ell>})$ for $\ell = 1, \ldots, d-1$ [170]. A tensor $\mathcal{T}$ with TT ranks at most $(R_1, \ldots, R_{d-1})$ is said to be represented in the TT format when it is given in terms of so-called TT cores $\mathcal{G}^{(1)} \in \mathbb{R}^{n_1 \times R_1}$, $\mathcal{G}^{(d)} \in \mathbb{R}^{R_{d-1} \times n_d}$ and $\mathcal{G}^{(\ell)} \in \mathbb{R}^{R_{\ell-1} \times n_\ell \times R_\ell}$ for $\ell = 2, \ldots, d-1$ as

$$\mathcal{T}_{i_1,\ldots,i_d} = \sum_{k_1=1}^{R_1} \sum_{k_1=1}^{R_1} \cdots \sum_{k_{d-1}=1}^{R_{d-1}} \mathcal{G}^{(1)}_{i_1,k_1} \mathcal{G}^{(2)}_{k_1,i_1,k_2} \cdots \mathcal{G}^{(d-1)}_{k_{d-2},i_{d-1},k_{d-1}} \mathcal{G}^{(d)}_{k_{d-1},i_d}, \tag{2.6}$$

for $i_\ell = 1, \ldots, n_\ell$, $\ell = 1, \ldots, d$. Storing the TT cores requires $\mathcal{O}((d-2)nR^2 + 2nR)$ storage under the assumption that $n_1 = n_2 = \cdots = n_d = n$ and $R_1 = R_2 = \cdots = R_{d-1} = R$. This results in linear (instead of exponential) growth with respect d under the (strong) assumption that R remains constant as d increases.

Remark 2.6. *Any tensor with multilinear ranks* $(r_1, \ldots, r_n)$ *can be represented in the Tucker format* (2.5)*. However, not every tensor that can be represented in Tucker format* (2.5) *with core tensor* $\mathcal{C} \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ *has multilinear ranks* $(r_1, \ldots, r_n)$*. The same applies for the TT ranks* $(R_1, \ldots, R_{d-1})$ *and the TT format* (2.6)*.*

Remark 2.7. *There exist many other low-rank tensor formats in literature including the hierarchical Tucker format [152], two-level approximations [186] and tensor networks [73, 247, 255]. For an in-depth overview we refer to the book by Hackbusch [154].*

2.1.4 Approximation algorithms

A classical algorithm to compute approximations of a given tensor $\mathcal{T}$ in the Tucker format (2.5) is the so-called higher order SVD (HOSVD) [88]. Its main idea is to compute the truncated SVDs of the matricizations $\mathcal{T}^{(\ell)}$ for $\ell = 1, \ldots, d$. The columns of the factor matrices $U^{(\ell)}$ are set to the leading left singular vectors of the corresponding SVD. Using orthogonal projections we obtain the core tensor analogous to the CUR decomposition (2.2) as

$$\mathcal{C} = \mathcal{T} \times_1 (U^{(1)})^T \times_2 (U^{(2)})^T \times \cdots \times (U^{(d)})^T. \tag{2.7}$$

Approximations in the TT format (2.6) can be computed using the so-called TT-SVD [249] by applying a several SVDs to $\mathcal{T}$. However, both the HOSVD and TT-SVD require the evaluation of the full tensor $\mathcal{T}$ to compute the approximation. This might not be feasible for large tensors. We refer to the survey [145] for an overview of algorithms to compute low-rank approximations of $\mathcal{T}$. In the following, we introduce how TT approximations can be computed efficiently using so-called TT-cross approximations [251] without evaluating every single entry of $\mathcal{T}$. For the efficient computation of Tucker approximations we refer to Chapter 3.

Cross approximation (2.1) can be generalized to tensors $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ in the following way. Let $\mathcal{I}^{\leq \ell} \subset \{(i_1, \ldots, i_\ell) | 1 \leq i_k \leq n_k, 1 \leq k \leq \ell\}$ and $\mathcal{I}^{>\ell} \subset \{(i_{\ell+1}, \ldots, i_d) | 1 \leq i_k \leq n_k, \ell + 1 \leq k \leq d\}$ denote index sets of cardinality R_ℓ for $\ell = 1, \ldots, d-1$. We obtain the approximation

$$\mathcal{T}_{i_1,\ldots,i_d} \approx \sum_{s_1=1}^{R_1} \sum_{t_1=1}^{R_1} \cdots \sum_{s_{d-1}=1}^{R_{d-1}} \sum_{t_{d-1}=1}^{R_{d-1}} \prod_{\ell=1}^{d} \mathcal{T}(\mathcal{I}^{\leq \ell-1}_{s_{\ell-1}}, i_\ell, \mathcal{I}^{>\ell}_{t_\ell}) (\mathcal{T}(\mathcal{I}^{\leq \ell}_{s_\ell}, \mathcal{I}^{>\ell}_{t_\ell}))^{-1}, \tag{2.8}$$

for $i_\ell = 1, \ldots, n_\ell$, $\ell = 1, \ldots, d$, where we use $\mathcal{I}^{\leq 0} = \mathcal{I}^{\leq d} = \mathcal{I}^{>d} = \{\}$. Note that Equation (2.8) is an approximation in TT format (2.6) [282]. Its cores can be computed by evaluating $\mathcal{O}(dnR^2)$ entries of $\mathcal{T}$ when $n_1 = \cdots = n_d = n$, $R_1 = \cdots = R_{d-1} = R$.

In order to derive suitable index sets, we present the rank-adaptive greedy restricted cross interpolation algorithm developed by Savostyanov [282]. We initialize the index sets $\mathcal{I}^{\leq \ell}, \mathcal{I}^{>\ell}$ randomly such that they each contain a single element and such that they are nested in the sense that $(i_1, \ldots, i_\ell) \in \mathcal{I}^{\leq \ell} \Rightarrow (i_1, \ldots, i_{\ell-1}) \in \mathcal{I}^{\leq \ell-1}$ and $(i_{\ell+1}, \ldots, i_d) \in \mathcal{I}^{>\ell} \Rightarrow (i_{\ell+2}, \ldots, i_d) \in \mathcal{I}^{>\ell+1}$. We now update these index sets in alternating order until the approximation (2.8) approximates $\mathcal{T}$ well. For this purpose so-called DMRG supercores [281] are formed. These are defined as subtensors $\mathcal{T}(\mathcal{I}^{\leq \ell-1}, :, :, \mathcal{I}^{>\ell+1})$ which are reshaped into tensors in $\mathbb{R}^{R_{\ell-1} \times n_\ell \times n_{\ell+1} \times R_{\ell+1}}$. The rank-adaptive algorithm proposed by Savostyanov [282, Algorithm 2] computes cross approximations (2.1) of the supercores of the form

$$\mathcal{T}(\mathcal{I}^{\leq \ell-1}, :, :, \mathcal{I}^{>\ell+1}) \approx \sum_{s_\ell=1}^{R_\ell} \sum_{t_\ell=1}^{R_\ell} \mathcal{T}(\mathcal{I}^{\leq \ell-1}, :, \mathcal{J}^{>\ell}_{t_\ell}) (\mathcal{T}(\mathcal{J}^{\leq \ell}_{s_\ell}, \mathcal{J}^{>\ell}_{t_\ell}))^{-1} \mathcal{T}(\mathcal{J}^{\leq \ell}_{s_\ell}, :, \mathcal{I}^{>\ell+1}),$$

where the index sets $\mathcal{J}^{\leq \ell}, \mathcal{J}^{\geq \ell}$ of cardinality at most $R_\ell + 1$ are constructed such that $\mathcal{I}^{\leq \ell} \subset \mathcal{J}^{\leq \ell}$ and $\mathcal{I}^{>\ell} \subset \mathcal{J}^{>\ell}$. We can now enhance the approximation (2.8) by replacing the index sets $\mathcal{I}^{\leq \ell}, \mathcal{I}^{>\ell}$ by $\mathcal{J}^{\leq \ell}, \mathcal{J}^{>\ell}$. In practice, this is achieved by determining the entry with the largest absolute value of the matrix

$$\left(\mathcal{T}(\mathcal{I}^{\leq \ell-1}, :, :, \mathcal{I}^{>\ell+1}) - \sum_{s_\ell=1}^{R_\ell} \sum_{t_\ell=1}^{R_\ell} \mathcal{T}(\mathcal{I}^{\leq \ell-1}, :, \mathcal{I}^{>\ell}_{t_\ell}) (\mathcal{T}(\mathcal{I}^{\leq \ell}_{s_\ell}, \mathcal{I}^{>\ell}_{t_\ell}))^{-1} \mathcal{T}(\mathcal{I}^{\leq \ell}_{s_\ell}, :, \mathcal{I}^{>\ell+1})\right)^{<2>} \tag{2.9}$$

using sampling as described in Remark 2.1. When the absolute value of this entry is above a prescribed tolerance, we obtain $\mathcal{J}^{\leq \ell}, \mathcal{J}^{\geq \ell}$ by adding the index corresponding to this entry to $\mathcal{I}^{\leq \ell}, \mathcal{I}^{\geq \ell}$. Note that this procedure adaptively increases the ℓth entry of the

TT rank of the approximation (2.8). Otherwise we set $\mathcal{J}^{\leq \ell}, \mathcal{J}^{\geq \ell}$ to $\mathcal{I}^{\leq \ell}, \mathcal{I}^{\geq \ell}$, i.e. we do not change the approximation. This process is repeatedly performed for $\ell = 1, \ldots, d-1$ and stopped once the error of the approximation (2.8) is sufficiently small at sample points. We formalize this procedure in Algorithm 3.

Algorithm 3 TT-cross

```
Input: procedure to evaluate entries of the tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, tolerance $\varepsilon$
Output: TT approximation (2.6) of $\mathcal{T}$ with core tensors $\mathcal{H}^{(\ell)}$
Initialize $\mathcal{I}^{\leq \ell}, \mathcal{I}^{> \ell}$ randomly with only one element each for $\ell = 1, \ldots, d-1$
while Error of the approximation (2.8) of $\mathcal{T}$ is larger than $\varepsilon$ at sample points.
    for $\ell = 1, \cdots, d-1$
        Find a large entry of the matrix (2.9) and update $\mathcal{I}^{\leq \ell}, \mathcal{I}^{> \ell}$ as described in Section 2.1.4.
Compute the TT cores $\mathcal{G}^{(\ell)}$ corresponding to the approximation (2.8).
```

2.2 Functional low-rank approximation

The concept of low-rank approximations can be generalized from tensor products of finite dimensional vector spaces to tensor products of infinite dimensional vector spaces [189]. Instead of matrices or tensors we are now interested in multivariate functions defined on the tensor product domain $[-1, 1]^d$. In this continuous setting, the pendant to rank-1 tensors are separable functions; see Equation (1.2). The rank r of a function $f : [-1, 1]^d \to \mathbb{R}$ is the minimal $r \in \mathbb{N} \cup \{\infty\}$ for which f can be written as a sum of separable functions, i.e.

$$f(x_1, x_2, \ldots, x_d) = \sum_{k=1}^{r} g_k^{(1)}(x_1) g_k^{(2)}(x_2) \cdots g_k^{(d)}(x_d), \qquad \text{for all } x_\ell \in [-1, 1],\ \ell = 1, \ldots, \ell, \tag{2.10}$$

where $g_k^{(\ell)} : [-1, 1] \to \mathbb{R}$ for $k = 1, \ldots, r,\ \ell = 1, \ldots, d$.

We are again interested in approximating $f : [-1, 1]^d \to \mathbb{R}$ by a function $h : [-1, 1]^d \to \mathbb{R}$ subject to the constraint that the rank of h is at most r for a fixed $r \in \mathbb{N}$. The function h is called (functional) low-rank approximation of f. Note that the original problem of finding separable decompositions of functions is intimately connected to low-rank decompositions of matrices and tensors [154, Chapter 7]. Throughout this work we typically measure the approximation error using the uniform norm $|| \cdot ||_\infty$ or the L^p-norm $|| \cdot ||_{L^p}$ defined on $[-1, 1]^d$. We want to emphasize that it is reasonable to search for low-rank approximations of f even when the rank of f is infinity. For example, truncating the Schmidt decomposition [285] of an arbitrary function in the Lebesgue space $L^2([-1, 1]^2)$ yields the best rank-r approximation with respect to the L^2-norm [147].

Analogous to Section 2.1.3, we can impose additional structure on the sum of separable functions (2.10). In the following, we generalize the Tucker format (2.5) and the TT format (2.6) from tensors to the continuous setting. This leads to functional low-rank approximations in the so-called functional Tucker [100, 163, 221, 270] and functional TT format [41, 139, 141].

Let $f : [-1,1]^d \to \mathbb{R}$. The multilinear rank $(r_1, \dots, r_d) \in (\mathbb{N} \cup \{\infty\})^d$ of f is defined as $r_\ell = \operatorname{rank}_{\{\ell\}}(f)$ for $\ell = 1, \dots, d$, where $\operatorname{rank}_{\{\ell\}}(f)$ denotes the minimum $r \in \mathbb{N} \cup \{\infty\}$ such that f can be written as in terms of functions $g_k : [-1,1] \to \mathbb{R}$ and $h_k : [-1,1]^{d-1} \to \mathbb{R}$ for $k = 1, \dots, r$ as

$$f(x_1, \dots, x_d) = \sum_{k=1}^{r} g_k(x_\ell) h_k(x_1, \dots, x_{\ell-1}, x_{\ell+1}, \dots, x_d).$$

A function $f : [-1,1]^d$ with multilinear rank at most $(r_1, \dots, r_d)$ is said to be represented in functional Tucker format when it is given in terms of a so-called a core tensor $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}$ and univariate functions $u_k^{(\ell)} : [-1,1] \to \mathbb{R}$ for $k = 1, \dots, r_\ell,\ \ell = 1, \dots, d$ as

$$f(x_1, \dots, x_d) = \sum_{k_1=1}^{r_1} \sum_{k_2=1}^{r_2} \cdots \sum_{k_d=1}^{r_d} \mathcal{C}_{k_1,k_2,\dots,k_d} u_{k_1}^{(1)}(x_1) u_{k_2}^{(2)}(x_2) \cdots u_{k_d}^{(d)}(x_d). \tag{2.11}$$

The TT rank $(R_1, \dots, R_{d-1}) \in (\mathbb{N} \cup \{\infty\})^{d-1}$ of f is defined as $R_\ell = \operatorname{rank}_{<\ell>}(f)$ for $\ell = 1, \dots, d-1$, where $\operatorname{rank}_{<\ell>}(f)$ denotes the minimum $R \in \mathbb{N} \cup \{\infty\}$ such that f can be written as in terms of functions $g_k : [-1,1]^\ell \to \mathbb{R}$ and $h_k : [-1,1]^{d-\ell} \to \mathbb{R}$ for $k = 1, \dots, R$ as

$$f(x_1, \dots, x_d) = \sum_{k=1}^{R} g_k(x_1, \dots, x_\ell) h_k(x_{\ell+1}, \dots, x_d).$$

A function $f : [-1,1]^d$ with TT ranks at most $(r_1, \dots, r_{d-1})$ is said to be represented in functional TT format when it is given in terms of univariate functions $g_{\alpha_1}^{(1)} : [-1,1] \to \mathbb{R}$ for $\alpha_1 = 1, \dots, R_1$, $g_{\alpha_{d-1}}^{(d)} : [-1,1] \to \mathbb{R}$ for $\alpha_{d-1} = 1, \dots, R_{d-1}$ and $g_{\alpha_{\ell-1},\alpha_\ell}^{(\ell)} : [-1,1] \to \mathbb{R}$ for $\alpha_\ell = 1, \dots, R_\ell,\ \ell = 2, \dots, d-1$ as

$$f(x_1, \dots, x_d) = \sum_{\alpha_1=1}^{R_1} \dots, \sum_{\alpha_{d-1}=1}^{R_{d-1}} g_{\alpha_1}^{(1)}(x_1) g_{\alpha_1,\alpha_2}^{(2)}(x_2) \cdots g_{\alpha_{d-2},\alpha_{d-1}}^{(d-1)}(x_{d-1}) g_{\alpha_{d-1}}^{(d)}(x_d). \tag{2.12}$$

Remark 2.8. *We want to emphasize that many other low-rank approximation formats can be generalized analogously from tensors to functions. For instance, generalizations of the hierarchical Tucker format are used in [21, 91, 117, 241, 242, 286].*

2.3 Chebyshev interpolation

2.3.1 Chebyshev interpolation for univariate functions.

In the following, we recall the fundamental ideas of Chebyshev interpolation.

Let $f : [-1, 1] \to \mathbb{R}$ and $n \in \mathbb{N}$. The Chebyshev interpolant p of f with degree n is the unique polynomial of degree n that interpolates f in the Chebyshev points (of the second kind) $x_k = \cos(k\pi/n),\ k = 0, \dots, n$, i.e. $p(x_k) = f(x_k)$ for $k = 0, \dots, n$. Using Chebyshev polynomials $T_k(x) = \cos(k \cos^{-1}(x)),\ k = 0, \dots, n$ [67] as basis for the vector space of polynomials of degree at most n, we can write the interpolant as

$$p(x) = \sum_{i=0}^{n} a_i T_i(x), \tag{2.13}$$

where $a_i \in \mathbb{R},\ i = 0, \dots, n$ denotes the so-called Chebyshev coefficients. These coefficients are uniquely determined by the interpolation property

$$f(x_k) = a_0 T_0(x_k) + a_1 T_1(x_k) + \cdots + a_n T_n(x_k), \qquad \text{for } k = 0, \dots, n.$$

These $n + 1$ equations form a linear system, which can be inverted. By using the following property of the Chebyshev polynomials

$$\frac{1}{2}\Big(T_i(x_0)T_j(x_0) + T_i(x_n)T_j(x_n)\Big) + \sum_{k=1}^{n-1} T_i(x_k)T_j(x_k) = \begin{cases} 0 & i \neq j, \\ \frac{n}{2} & 0 < i = j < n, \\ n & i = j = 0 \text{ or } i = j = n, \end{cases}$$

we can construct the matrix $F \in \mathbb{R}^{(n+1)\times(n+1)}$ defined as

$$F = \frac{2}{n} \begin{pmatrix} \frac{1}{4}T_0(x_0) & \frac{1}{2}T_0(x_1) & \frac{1}{2}T_0(x_2) & \dots & \frac{1}{4}T_0(x_n) \\ \frac{1}{2}T_1(x_0) & T_1(x_1) & T_1(x_2) & \dots & \frac{1}{2}T_1(x_n) \\ \frac{1}{2}T_2(x_0) & T_2(x_1) & T_2(x_2) & \dots & \frac{1}{2}T_2(x_n) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{4}T_n(x_0) & \frac{1}{2}T_n(x_1) & \frac{1}{2}T_n(x_2) & \dots & \frac{1}{4}T_n(x_n) \end{pmatrix}, \tag{2.14}$$

which maps the vector of function evaluations $(f(x_0), f(x_1), \dots, f(x_n))$ onto the vector of Chebyshev coefficients $(a_0, a_1, \dots, a_n)$.

The following theorem summarizes the error bounds in [322, Theorem 8.2] and [322, Theorem 7.1]. Note that the error bound decays exponentially with respect to n in part a) and geometrically in part b).

Theorem 2.1. *a) Suppose that the continuous function $f : [-1, 1] \to \mathbb{R}$ can be extended to an analytic function f^* on E_ρ with $\rho > 1$, where E_ρ denotes the Bernstein ellipse, i.e. the closed ellipse with foci at ± 1 and the sum of major and minor semi-axes equal* *to ρ. Then the Chebyshev interpolant p of f with degree n satisfies*

$$\|f-p\|_\infty \leq \frac{4\rho^{-n}}{\rho-1}\max_{z\in E_\rho}|f^*(z)|,$$

b) Assume that the function $f:[-1,1]\to\mathbb{R}$ has $(k-1)$ absolutely continuous derivatives and assume that the total variation of its kth derivative is bounded by V. Then the Chebyshev interpolant p of f with degree $n>k$ satisfies

$$\|f-p\|_\infty \leq \frac{4V}{\pi k(n-k)^k}.$$

Remark 2.9. *Throughout this work, we follow the standard notation established in [240], where the Chebyshev points, polynomials and coefficients are indexed starting from 0. Other vectors, matrices and tensors are indexed starting from 1.*

Remark 2.10. *Given a function $f:[-1,1]\to\mathbb{R}$, the cosine transform computes the values*

$$\tilde{a}_k = f(x_0)T_k(x_0) + f(x_n)T_k(x_n) + 2\sum_{i=1}^{n-1} f(x_i)T(x_i), \qquad \textit{for } k=0,\dots,n.$$

This can be written equivalently in terms of a discrete Fourier transform [135] as

$$\tilde{a}_k = \sum_{j=0}^{2n-1} f(\cos(\frac{\pi j}{n}))\exp(\frac{2\pi ijn}{2n}), \qquad \textit{for } k=0,\dots,n,$$

where i denotes the imaginary unity. Note that $a_0 = 1/2\cdot\tilde{a}_0$, $a_n = 1/2\cdot\tilde{a}_n$ and $a_k = \tilde{a}_k$ for $k=1,\dots,n-1$. Thus, we can use the fast Fourier transform to compute the coefficient vector in a fast and stable way.

Remark 2.11. *Let $m\in\mathbb{N}$. Note that the Chebyshev points for $n=m$ are nested within the Chebyshev points for $n=2m$. This implies, that given the function evaluations to compute a degree m interpolant, we only need to evaluate the function at m additional points to compute the degree $2m$ interpolant. Moreover, there exist heuristics such as chop in Chebfun [16] to estimate whether a given Chebyshev interpolant is an accurate approximation of f. We can efficiently determine a sufficiently large degree for an accurate interpolant by initially starting with a small degree and by doubling the degree until the heuristic states that the interpolation is accurate.*

2.3.2 Tensorized Chebyshev interpolation for multivariate functions

Chebyshev interpolation can be extended to multivariate functions using tensorization [237]. Let $f:[-1,1]^d\to\mathbb{R}$ and $(n_1,n_2,\dots,n_d)\in\mathbb{N}^d$. The Chebyshev interpolant p of f with degree $(n_1,n_2,\dots,n_d)$ is the unique multivariate polynomial of degree

$(n_1, n_2, \dots, n_d)$, which satisfies

$$p(x_{i_1}^{(1)}, x_{i_2}^{(2)}, \dots, x_{i_d}^{(d)}) = f(x_{i_1}^{(1)}, x_{i_2}^{(2)}, \dots, x_{i_d}^{(d)}), \quad \text{for } i_\ell = 0, \dots, n_\ell,\ \ell = 1, \dots, d, \tag{2.15}$$

where the Chebyshev points are denoted by $x_k^{(\ell)} = \cos(k\pi/n_\ell),\ k = 0, \dots, n_\ell$. We can write the interpolant using a basis containing tensorized Chebyshev polynomial as

$$p(x_1, x_2, \dots, x_d) = \sum_{i_1=0}^{n_1} \sum_{i_2=0}^{n_2} \dots \sum_{i_d=0}^{n_d} \mathcal{A}_{i_1,i_2,\dots,i_d} T_{i_1}(x_1) T_{i_2}(x_2) \dots T_{i_d}(x_d), \tag{2.16}$$

where $\mathcal{A} \in \mathbb{R}^{(n_1+1)\times(n_2+1)\times\dots\times(n_d+1)}$ denotes the so-called coefficient tensor. Analogously to the case of univariate interpolation, the coefficient tensor is uniquely determined by the interpolation property (2.15), which can be written as

$$\mathcal{T}_{j_1,j_2,\dots,j_d} = \sum_{i_1=0}^{n_1} \sum_{i_2=0}^{n_2} \dots \sum_{i_d=0}^{n_d} \mathcal{A}_{i_1,i_2,\dots,i_d} T_{i_1}(x_{j_1}^{(1)}) T_{i_2}(x_{j_2}^{(2)}) \dots T_{i_d}(x_{j_d}^{(d)}), \tag{2.17}$$

for all $j_\ell = 0, \dots, n_\ell$, $\ell = 1, \dots, d$, where the evaluation tensor $\mathcal{T} \in \mathbb{R}^{(n_1+1)\times(n_2+1)\times\dots\times(n_d+1)}$ is defined elementwise as

$$\mathcal{T}_{j_1,j_2,\dots,j_d} = f(x_{j_1}^{(1)}, x_{j_2}^{(2)}, \dots, x_{j_d}^{(d)}). \tag{2.18}$$

Let $F^{(\ell)} \in \mathbb{R}^{(n_\ell+1)\times(n_\ell+1)}$ denote the matrix F defined in (2.14) with n replaced by n_ℓ. Multiplying the system (2.17) with $F^{(1)}$ in the first mode yields

$$\sum_{i_1=0}^{n_1} F_{j_1,i_1}^{(1)} \mathcal{T}_{i_1,j_2,\dots,j_d} = \sum_{i_2=0}^{n_2} \sum_{i_3=0}^{n_3} \dots \sum_{i_d=0}^{n_d} \mathcal{A}_{j_1,i_2\dots,i_d} T_{i_2}(x_{j_2}^{(2)}) T_{i_3}(x_{j_3}^{(3)}) \dots T_{i_d}(x_{j_d}^{(d)})$$

for all $j_\ell = 0, \dots, n_\ell$, $\ell = 1, \dots, d$. Repeating this procedure in each mode yields the explicit expression for the coefficient tensor

$$\mathcal{A} = \mathcal{T} \times_1 F^{(1)} \times_2 F^{(2)} \times_3 \cdots \times_d F^{(d)}. \tag{2.19}$$

We would like to point out that the mode-ℓ product of the matrix $F^{(\ell)}$ and $\mathcal{T}$ can be computed efficiently by using the fast Fourier transform described in Remark 2.9. Note that computation of the coefficient tensor (2.19) requires the evaluation of $(n_1+1)\cdot(n_2+1)\cdots(n_d+1)$ points of f.

The following theorem restates the error bound in [280, Lemma 7.3.3.]. Note that the approximation error (2.20) decays exponentially with respect to $\min\{n_1, \dots, n_d\}$ for analytic functions f.

Theorem 2.2. *Suppose that the continuous function $f : [-1,1]^d \to \mathbb{R}$ can be extended to an analytic function f^* on $\mathcal{E}_\rho = E_{\rho_1} \times E_{\rho_2} \times \dots \times E_{\rho_d}$ with $\rho_\ell > 1$ for $\ell = 1, \dots, d$, where E_ρ again denotes the Bernstein ellipse defined in Theorem 2.1. Then the Chebyshev* *interpolant p of f with degree $(n_1, \dots, n_d)$ satisfies*

$$\|f - p\|_\infty \leq 2^{1+d/2}\sqrt{d}\rho_{\min}^{-n_{\min}} \left(1 - \rho_{\min}^{-2}\right)^{-d/2} \max_{z \in \mathcal{E}_\rho} |f^*(z)|, \tag{2.20}$$

where $\rho_{\min} = \min\{\rho_1, \rho_2, \dots, \rho_d\}$ and $n_{\min} = \min\{n_1, n_2, \dots, n_d\}$.

2.3.3 Coefficient approximation

Due to the curse of dimensionality, it might not be feasible to compute the coefficient tensor (2.19) for large d. In the following lemma, we study how approximations of the evaluation tensor (2.18) affect the interpolant (2.16). Similar results with respect to the L^2-norm can be found in [41].

Lemma 2.2. *Let p be defined as in (2.16). For $\hat{\mathcal{T}} \in \mathbb{R}^{(n_1+1)\times(n_2+1)\times\cdots\times(n_d+1)}$, we define the polynomial*

$$\hat{p}(x_1, \dots, x_n) = \sum_{i_1=0}^{n_1} \cdots \sum_{i_d=0}^{n_d} \hat{\mathcal{A}}_{i_1,\dots,i_d} T_{i_1}(x_1) \cdots T_{i_d}(x_d), \tag{2.21}$$

where

$$\hat{\mathcal{A}} = \hat{\mathcal{T}} \times_1 F^{(1)} \times_2 F^{(2)} \times_3 \cdots \times_d F^{(d)}. \tag{2.22}$$

Then

$$\|f - \hat{p}\|_\infty \leq \|f - p\|_\infty + \prod_{\ell=1}^{d} \Big(\frac{2}{\pi}\log(n_\ell) + 1\Big) \|\mathcal{T} - \hat{\mathcal{T}}\|_\infty. \tag{2.23}$$

Proof. By applying the triangle inequality we obtain

$$\|f - \hat{p}\|_\infty \leq \|f - p\|_\infty + \|p - \hat{p}\|_\infty.$$

The function $\tilde{p} = p - \hat{p}$ is the unique polynomial of degree n satisfying

$$\tilde{p}(x_{i_1}^{(1)}, x_{i_2}^{(2)}, \dots, x_{i_d}^{(d)}) = (\mathcal{T} - \hat{\mathcal{T}})_{i_1 i_2 \dots i_d}, \qquad \text{for } i_\ell = 0, \dots, n_\ell,\ \ell = 1, \dots, n.$$

Let $n \in \mathbb{N}$. The Lebesgue constant Λ_n bounds the ratio of the uniform norm approximation error for univariate interpolation and the maximum absolute value at the $n{+}1$ interpolation nodes. For univariate Chebyshev interpolation we have $\Lambda_n \leq (2/\pi)\log(n) + 1$ [322]. In [223], it is shown that the product of the Lebesgue constants for univariate interpolation bounds this ratio for multivariate interpolation, i.e.

$$\|p - \hat{p}\|_\infty = \|\tilde{p}\|_\infty \leq \Lambda_{n_1}\Lambda_{n_2} \cdots \Lambda_{n_d} \|\mathcal{T} - \hat{\mathcal{T}}\|_\infty.$$

□

Note that the term $\|f-p\|_\infty$ in the error bound (2.23) does not depend on the approximation $\hat{\mathcal{T}}$. For analytic f, Theorem 2.2 states that we can choose a sufficiently large degree $(n_1, n_2, \dots, n_d)$ such that $\|f-p\|_\infty$ is arbitrarily small.

Remark 2.12. *In Lemma 2.2, we bound the error of* $\|p-\hat{p}\|_\infty$ *in terms of the Lebesgue constants and* $\|\mathcal{T}-\hat{\mathcal{T}}\|_\infty$*. In the following, we briefly discuss how* $\|p-\hat{p}\|_{L^1}$ *is related to a weighted norm of* $\mathcal{T}-\hat{\mathcal{T}}$*.*

The tensorized Clenshaw-Curtis quadrature formula [74, 86] yields

$$\begin{aligned}\|p-\hat{p}\|_{L^1} &= \int_{-1}^{1}\int_{-1}^{1}\cdots\int_{-1}^{1}|p(x_1,\dots,x_d)-\hat{p}(x_1,\dots,x_d)|dx_1\dots dx_d\\ &= \sum_{i-1=0}^{n_1}\sum_{i_2=0}^{n_2}\cdots\sum_{i_d=0}^{n_d}\mathcal{W}_{i_1,i_2,\dots,i_d}|T_{i_1,i_2,\dots,i_d}-\hat{T}_{i_1 i_2\dots i_d}|,\end{aligned}$$

where $\mathcal{W}_{i_1,i_2,\dots,i_d} = w^{(1)}_{i_1}w^{(2)}_{i_2}\cdots w^{(d)}_{i_d}$ *with nonnegative weights* $w^{(\ell)}_{i_\ell}$ *are defined for* $i_\ell = 0,\dots,n_\ell$, $\ell = 1,\dots,d$ *as in [334]*

$$\begin{aligned} w^{(\ell)}_{i_\ell} &= \frac{c^{(\ell)}_\ell}{(n_\ell)}\Big(1-\sum_{j=1}^{\lfloor n_\ell/2\rfloor}\frac{b^{(\ell)}_j}{4j^2-1}\cos\Big(\frac{2ji_\ell\pi}{n_\ell}\Big)\Big),\\ c^{(\ell)}_{i_\ell} &= \begin{cases}1 & i_\ell = 0 \text{ or } i_\ell = n_\ell,\\ 2 & \text{otherwise,}\end{cases}\\ b^{(\ell)}_{i_\ell} &= \begin{cases}1 & i_\ell = \frac{n_\ell}{2},\\ 2 & \text{otherwise.}\end{cases}\end{aligned}$$

Based on these weights, we define the weighted tensor seminorm

$$\|\mathcal{T}\|_{\mathcal{W}} = \sum_{i-1=0}^{n_1}\sum_{i_2=0}^{n_2}\cdots\sum_{i_d=0}^{n_d}\mathcal{W}_{i_1,i_2,\dots,i_d}|T_{i_1,i_2,\dots,i_d}|.$$

This yields the equality $\|p-\hat{p}\|_{L^1} = \|\mathcal{T}-\hat{\mathcal{T}}\|_{\mathcal{W}}$*. Note that best approximations* $\mathcal{T}^*_{\mathcal{W}}$ *of* $\mathcal{T}$ *with respect to the seminorm* $\|\cdot\|_{\mathcal{W}}$ *are related to best approximations* $\mathcal{T}^*_1$ *of* $\mathcal{W}*\mathcal{T}$ *in the* ℓ^1*-norm, where* $*$ *denotes the elementwise/Hadamard product for tensors. It holds*

$$(\mathcal{T}^*_{\mathcal{W}})_{i_1,\dots,i_d} = \begin{cases}(\mathcal{T}^*_1)_{i_1,\dots,i_d}/\mathcal{W}_{i_1,\dots,i_d} & \mathcal{W}_{i_1,\dots,i_d}\neq 0\\ (\mathcal{T}^*_1)_{i_1,\dots,i_d} & \mathcal{W}_{i_1,\dots,i_d} = 0.\end{cases}$$

for $i_\ell = 0,\dots,n_\ell$, $\ell = 1,\dots,d$.

2.3.4 Connection to functional low-rank approximations

Assume that $\hat{\mathcal{T}}$ in (2.21) is given in Tucker format (2.5) with multilinear rank $(r_1, \dots, r_d)$ as

$$\hat{\mathcal{T}} = \mathcal{C} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 \cdots \times_d U^{(d)}.$$

In this case the coefficient tensor $\hat{A}$ defined in (2.22) can be represented in the same low-rank format. This leads to a polynomial approximation (2.21) of the form

$$\begin{aligned} \hat{p}(x_1, \dots, x_d) = \sum_{i_1=1}^{r_1} \cdots \sum_{i_d=1}^{r_d} \mathcal{C}_{i_1,\dots,i_d} \Big(\sum_{k_1=0}^{n_1} \sum_{j_1=0}^{n_1} F^{(1)}_{k_1,j_1} U^{(1)}_{j_1,i_1} T_{k_1}(x_1) \Big) \cdots \\ \Big(\sum_{k_d=0}^{n_d} \sum_{j_d=0}^{n_d} F^{(d)}_{k_d,j_d} U^{(d)}_{j_d,i_d} T_{k_d}(x_d) \Big). \qquad (2.24) \end{aligned}$$

Note that this is an approximation in the functional Tucker format (2.11) with univariate functions

$$u_i^{(\ell)}(x_\ell) = \sum_{k=0}^{n_\ell} \sum_{j=0}^{n_\ell} F^{(\ell)}_{k,j} U^{(\ell)}_{j,i} T_k(x_\ell), \qquad \text{for } i = 1, \dots, r_\ell, \ \ell = 1, \dots, d.$$

The function $u_i^{(\ell)}(x_\ell)$ is the univariate Chebyshev interpolant based on the values stored in the ith colum of $U^{(\ell)}$.

Analogously, we can transform a tensor train approximation (2.6) of $\hat{\mathcal{T}}$ with TT cores $\mathcal{G}^{(\ell)}$ into a functional tensor train approximation (2.12) of f by defining the univariate functions

$$\begin{aligned} g^{(1)}_{\alpha_1}(x_1) &= \sum_{j=0}^{n_1} \sum_{k=0}^{n_1} F^{(1)}_{j,k} \mathcal{G}^{(1)}_{k,\alpha_1} T_j(x_1), \\ g^{(d)}_{\alpha_{d-1}}(x_d) &= \sum_{j=0}^{n_d} \sum_{k=0}^{n_d} F^{(d)}_{j,k} \mathcal{G}^{(d)}_{\alpha_{d-1},k} T_j(x_d), \\ g^{(\ell)}_{\alpha_{\ell-1},\alpha_\ell}(x_\ell) &= \sum_{j=0}^{n_\ell} \sum_{k=0}^{n_\ell} F^{(\ell)}_{j,k} \mathcal{G}^{(\ell)}_{\alpha_{\ell-1},k,\alpha_\ell} T_j(x_\ell). \end{aligned}$$

These can again be seen as univariate Chebyshev interpolants based on the values stored in the TT cores.

2.3.5 Proof of concept

The error bound (2.23) indicates that we ideally choose a Tucker approximation $\hat{\mathcal{T}}$ such that $\|\mathcal{T} - \hat{\mathcal{T}}\|_\infty$ is of the same order of magnitude as $\|f - p\|_\infty$. In the following, we demonstrate that there exist functions for which we can balance the contributions of these errors with $r_\ell \ll n_\ell$ for $\ell = 1, \dots, d$. In this case the approximation (2.24) is almost as accurate as the interpolant (2.16), but it requires significantly less storage.

We consider the function

$$f_\varepsilon(x, y, z) = \frac{1}{x + y + z + 3 + \varepsilon}$$

on $[-1, 1]^3$ with parameter $\varepsilon > 0$. Let $\tau \geq 0$. In the following paragraphs, we show that the Chebyshev interpolant p_ε defined in (2.16) requires polynomial degrees $n_\ell - 1 = \mathcal{O}(1/\log(1 + \sqrt{\varepsilon}))$ to achieve an accuracy of $\|p_\varepsilon - f_\varepsilon\|_\infty \leq \tau$. However, one can achieve $\|\mathcal{T} - \hat{\mathcal{T}}\|_\infty \leq \tau$ with multilinear ranks $r_\ell \leq \mathcal{O}(|\log(\varepsilon)|)$, which grows much slower than $\mathcal{O}(1/\log(1 + \sqrt{\varepsilon})) \approx \mathcal{O}(\varepsilon^{-1/2})$ for $\varepsilon \to 0$. For small values of ε this implies that we can balance the error contributions with $r_\ell \ll n_\ell$ for $\ell = 1, \dots, d$.

Polynomial Degree For the degree (n, n, n) Chebyshev interpolant p_ε defined in (2.16) we require $\|p_\varepsilon - f_\varepsilon\|_\infty \leq \tau$, which is equivalent to $\|\varepsilon(p_\varepsilon - f_\varepsilon)\|_\infty \leq \varepsilon\tau$. By Theorem 2.2,

$$\|\varepsilon(p_\varepsilon - f_\varepsilon)\|_\infty \leq \mathcal{O}(\rho_{\min}^{-n} \cdot \max_{(x,y,z)\in\mathcal{E}_\rho} |\varepsilon f_\varepsilon^*(x, y, z)|).$$

We set $\rho_1 = \rho_2 = \rho_3 = 1 + \varepsilon/6 + \sqrt{(1 + \varepsilon/6)^2 - 1}$ and extend $\varepsilon f_\varepsilon$ analytically to $\varepsilon f_\varepsilon^*(x, y, z) = \varepsilon\,(x + y + z + 3 + \varepsilon)^{-1}$ on $\mathcal{E}_\rho$. By construction $\max_{(x,y,z)\in\mathcal{E}_\rho} |\varepsilon f_\varepsilon^*(x, y, z)| = 2$ is assumed for $x = y = z = -1 - \varepsilon/6$, where $|x + y + z + 3 + \varepsilon|$ is minimized. Hence, we can choose $n = \mathcal{O}(1/\log(1 + \sqrt{\varepsilon}))$ to obtain the desired accuracy. Although this is only an upper bound for the polynomial degree required, numerical experiments reported below indicate that it is tight.

Multilinear Rank An a priori approximation with exponential sums is used to obtain a bound on the multilinear rank for a tensor containing function values of f_ε; see [153]. Given $R > 1$ and $r \in \mathbb{N}$, Braess and Hackbusch [47] showed that there exist coefficients a_i and b_i such that

$$\left| \frac{1}{x} - \sum_{i=1}^{r} a_i \exp(-b_i x) \right| \leq 16 \exp\left(-\frac{r\pi^2}{\log(8R)} \right), \quad \forall x \in [1, R]. \tag{2.25}$$

Trivially, we have $\varepsilon f_\varepsilon(x, y, z) = 1/\omega$ for the substitution $\omega = (x + y + z + 3 + \varepsilon)/\varepsilon$ with $\omega \in [1, 1 + 6/\varepsilon]$. Applying (2.25) yields that there exist a_i and b_i such that $|1/\omega - \sum_{i=1}^{r} a_i \exp(-b_i\omega)| \leq \tau\varepsilon$ or, equivalently,

$$||f_\varepsilon(x, y, z) - g_\varepsilon(x, y, z)||_\infty \leq \tau \tag{2.26}$$

for every $x, y, z \in [-1, 1]$ when

$$r \geq \frac{-\log\left(8\left(1+\frac{6}{\varepsilon}\right)\right)\log\left(\frac{\tau}{16}\right)}{\pi^2} = \mathcal{O}(|\log(\varepsilon)|),$$

where

$$g_\varepsilon(x, y, z) = \sum_{i=1}^{r} \frac{a_i}{\varepsilon} \cdot \exp\left(-\frac{b_i}{\varepsilon}x\right) \cdot \exp\left(-\frac{b_i}{\varepsilon}y\right) \cdot \exp\left(-\frac{b_i}{\varepsilon}z\right) \cdot \exp\left(-\frac{b_i}{\varepsilon}(3+\varepsilon)\right).$$

The approximation g_ε in (2.26) has multilinear rank (r, r, r). In turn, the tensor $\hat{\mathcal{T}}_{i,j,k} = g_\varepsilon(x_i^{(1)}, x_j^{(2)}, x_k^{(3)})$ has multilinear rank at most (r, r, r) and satisfies $\|\mathcal{T} - \hat{\mathcal{T}}\|_\infty \leq \tau$.

Comparison In Figure 2.2, we estimate the maximal polynomial degree required to compute a Chebyshev interpolant with accuracy $\tau = 10^{-10}$ for selected fibers of f_ε, which is a lower bound for the required polynomial degrees n_ℓ. It perfectly matches the asymptotic behavior of the derived upper bound $\mathcal{O}(1/\log(1+\sqrt{\varepsilon}))$. In Figure 2.2, we also plot the multilinear ranks from the truncated HOSVD [88] with tolerance τ applied to the tensor containing the evaluation of f_ε on a $150 \times 150 \times 150$ Chebyshev grid. This estimate serves as a lower bound for the multilinear rank required to approximate f_ε. Due to the limited grid size, this estimate does not fully match the asymptotic behavior $|\log(\varepsilon)|$, but nonetheless it clearly reflects that the multilinear ranks can be much smaller than the polynomial degrees, as predicted by $|\log(\varepsilon)| \ll 1/\log(1+\sqrt{\varepsilon})$, for sufficiently small ε.

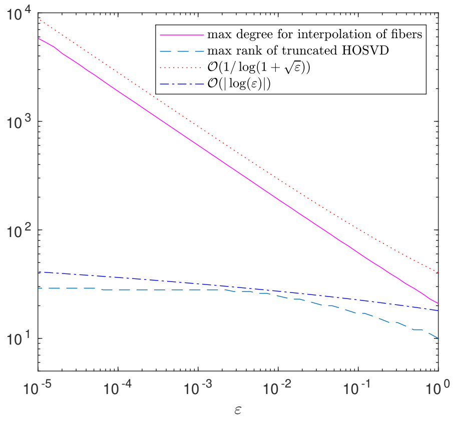


Figure 2.2 – Comparison of the theoretical upper bounds $\mathcal{O}(|\log(\varepsilon)|)$ for the multilinear rank and $\mathcal{O}(1/\log(1+\sqrt{\varepsilon}))$ for the polynomial degree for varying ε, the maximal polynomial degree, used by Chebfun to approximate selected functions fibers of f_ε up to accuracy 10^{-10}, and the maximal multilinear rank of the truncated HOSVD with tolerance 10^{-10} of a sample tensor on a $150 \times 150 \times 150$ Chebyshev grid. The constants in the bounds are chosen to result in curves close to the data.

3 Functional Tucker approximation

In this chapter, we focus on developing an efficient algorithm to approximate trivariate functions (that is, functions depending on three variables) defined on the tensor-product domain $[-1,1]^3$ in the functional Tucker format (2.11). We proceed by combining tensorized Chebyshev interpolation (2.16) with a Tucker decomposition (2.5) of the coefficient tensor (2.19). This leads to a functional Tucker approximation of the form (2.24).

Hashemi and Trefethen followed this approach to develop an approximation algorithm for Chebfun3 [163]. Their algorithm is adaptive in both the multilinear rank and the polynomial degree of the computed approximation, but the construction of the approximation proceeds indirectly, via a so-called slice-Tucker decomposition of the evaluation tensor (2.18). As a consequence, Chebfun3 sometimes uses unnecessarily many function evaluations and does not fully benefit from the potential of the Tucker decomposition to reduce, sometimes dramatically, the computational cost.

We propose a novel algorithm that utilizes univariate fibers instead of bivariate slices to construct a Tucker decomposition (2.5) of the evaluation tensor (2.18) directly. We call our approach Chebfun3F to emphasize that it is based on selecting the *Fibers* of the evaluation tensor to construct the initial factor matrices. In a second step, we replace orthogonal projections (2.7) by oblique projections (2.3) based on DEIM to construct the core tensor. We combine this approach with heuristics similar to the ones used in Chebfun3 for choosing the polynomial degree and for the accuracy verification. Chebfun3F reduces the cost for the approximation in terms of the number of function evaluations for nearly all functions considered, typically by 75%, and sometimes by over 98%.

This chapter is based on the article [100]. Its remainder is structured as follows. In Section 3.1, we briefly recall the approximation algorithm currently used in Chebfun3. Section 3.2 introduces our novel algorithm Chebfun3F. In Section 3.3, we perform numerical experiments to compare Chebfun3, Chebfun3F and sparse grid interpolation. We then analyse the coefficient decay of the resulting approximations and discuss how Chebfun3F can be incorporated into Chebfun in Section 3.4. In Section 3.5, we analyze further properties of functional Tucker decompositions computed based on tensorized Chebyshev interpolation.

3.1 Existing algorithm: Chebfun3

In this section, we recall how an approximation of the form (2.11) is computed in Chebfun3 [163]. As discussed in Section 2.3.5, there are often situations in which approximations (2.24) require a much smaller $(r_1, \dots, r_d)$ compared to the polynomial degree $(n_1, \dots, n_d)$. Chebfun3 benefits from such a situation by first using a coarse evaluation tensor $\mathcal{T}_c$ to identify the fibers needed for the low-rank approximation. This allows to construct the actual approximation from a finer sample tensor $\mathcal{T}_f$ by only evaluating these fibers instead of the whole tensor.

Chebfun3 consists of three phases: preparation of the approximation by identifying fibers for a so-called block term decomposition [87] of $\mathcal{T}_c$, refinement of the fibers, conversion and compression of the refined block term decomposition of $\mathcal{T}_f$ into functional Tucker format (2.11).

3.1.1 Phase 1: Block term decomposition

In Chebfun3, the coarse evaluation tensor $\mathcal{T}_c \in \mathbb{R}^{(n_1^{(c)}+1)\times(n_2^{(c)}+1)\times(n_3^{(c)}+1)}$ is initially obtained by sampling f on a $17 \times 17 \times 17$ grid of Chebyshev points. A block term decomposition of $\mathcal{T}_c$ is obtained by applying ACA (see Algorithm 1) recursively. In the first step, ACA is applied to a matricization of $\mathcal{T}_c$, say, the mode-1 matricization $\mathcal{T}_c^{\{1\}}$. This results in index sets I, J such that

$$\mathcal{T}_c^{\{1\}} \approx \mathcal{T}_c^{\{1\}}(:, J) \left(\mathcal{T}_c^{\{1\}}(I, J)\right)^{-1} \mathcal{T}_c^{\{1\}}(I, :), \tag{3.1}$$

where $\mathcal{T}_c^{\{1\}}(:, J)$ contains mode-1 fibers of $\mathcal{T}_c$ and $\mathcal{T}_c^{\{1\}}(I, :)$ contains mode-$(2, 3)$ slices of $\mathcal{T}_c$. For each $i \in I$, such a slice $\mathcal{T}_c^{\{1\}}(i, :)$ is reshaped into a matrix $S_i = \mathcal{T}_c(i, :, :) \in \mathbb{R}^{(n_2^{(c)}+1)\times(n_3^{(c)}+1)}$ and, in the second step, approximated by again applying ACA:

$$S_i \approx S_i(:, L_i) \left(S_i(K_i, L_i)\right)^{-1} S_i(K_i, :), \tag{3.2}$$

where $S_i(:, L_i)$ and $S_i(K_i, :)$ contain mode-2 and mode-3 fibers of $\mathcal{T}_c$, respectively. Combining (3.1) and (3.2) yields the approximation

$$\mathcal{T}_c^{\{1\}} \approx \mathcal{T}_c^{\{1\}}(:, J) \left(\mathcal{T}_c^{\{1\}}(I, J)\right)^{-1} \begin{pmatrix} \mathsf{vec}(S_1(:, L_1) \left(S_1(K_1, L_1)\right)^{-1} S_1(K_1, :)) \\ \mathsf{vec}(S_2(:, L_2) \left(S_2(K_2, L_2)\right)^{-1} S_2(K_1, :)) \\ \vdots \end{pmatrix}, \tag{3.3}$$

where vec denotes vectorization, i.e. the reshaping of a matrix into a vector. Reshaping this approximation into a tensor can be viewed as a block term decomposition in the sense of [87, Definition 2.2.]. This approximation scheme is visualized in Figure 3.1.

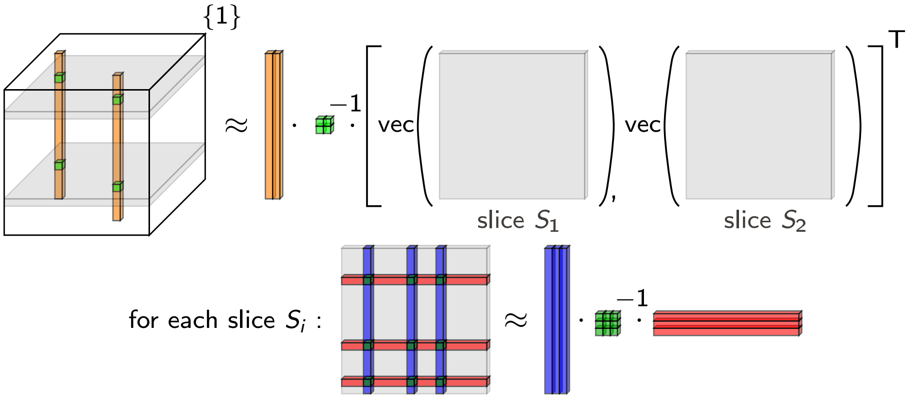


Figure 3.1 – Visualization of a block term decomposition (3.3).

If the ratios of $|I|/(n_1^{(c)}+1)$, $|K_i|/(n_2^{(c)}+1)$ and $|L_i|/(n_3^{(c)}+1)$ are larger than the heuristic threshold $(2\sqrt{2})^{-1}$ the coarse grid resolution $(n_1^{(c)}+1, n_2^{(c)}+1, n_3^{(c)}+1)$ is deemed insufficient to identify fibers. If this is the case, $n_\ell^{(c)}$ is increased to $\left\lfloor \sqrt{2}^{\left\lfloor 2\log_2(n_\ell^{(c)}+1)+1 \right\rfloor} \right\rfloor$ and Phase 1 is repeated.

3.1.2 Phase 2: Refinement

The block term decomposition (3.3) is composed of fibers of $\mathcal{T}_c$. Such a fiber $\mathcal{T}_c(:,j,k)$ corresponds to the evaluation of a univariate function $f(\cdot,y,z)$ for certain fixed y, z. We apply the Chebfun heuristic described in Remark 2.11 to decide whether the function values in $\mathcal{T}_c(:,j,k)$ suffice to yield an accurate interpolation of $f(\cdot,y,z)$ [16]. If this is not the case, we add nested evaluations of $f(\cdot,y,z)$ to refine the approximation.

In Chebfun3 this heuristic is applied to all mode-ℓ fibers contained in (3.3) simultaneously. Let $n_\ell^{(f)}$ denote the polynomial degree for which the heuristic states that the interpolation of is accurate. The refined mode-ℓ fibers are now vectors of length $n_\ell^{(f)}+1$. Replacing all fibers in (3.3) by their refined counterparts yields an approximation of the tensor $\mathcal{T}_f$, which contains the evaluations of f on a $(n_1^{(f)}+1) \times (n_2^{(f)}+1) \times (n_3^{(f)}+1)$ Chebyshev grid. Note that $\mathcal{T}_f$ might be very large and is never computed explicitly. We visualize the refinement idea in Figure 3.2.

3.1.3 Phase 3: Compression

In the third phase of the Chebfun3 constructor, the refined block term decomposition is converted and compressed to the desired Tucker format (2.11), where the interpolants

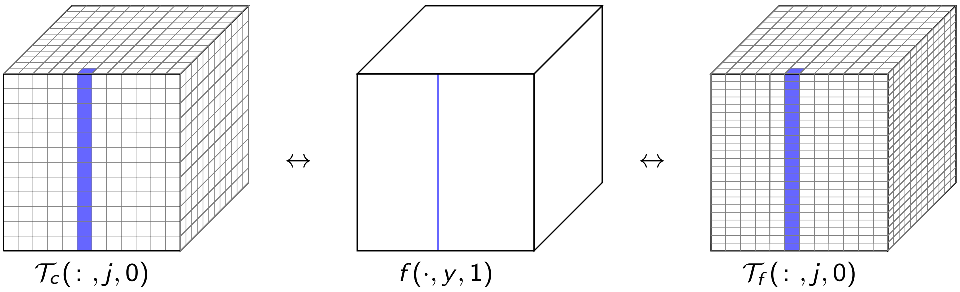


Figure 3.2 – Visualization of the refinement step described in Section 3.1.2. In the refinement step, we replace the initial fiber $\mathcal{T}_c(:,j,0)$ by the fiber $\mathcal{T}_f(:,j,0)$. Here we depict tensors for $(n_1^{(c)}, n_2^{(c)}, n_3^{(c)}) = (11, 11, 11)$ and $(n_1^{(f)}, n_2^{(f)}, n_3^{(f)}) = (22, 11, 11)$.

$u_i(x)$ are stored as Chebfun objects [27]; see [163] for details. Lemma 2.2 guarantees a good approximation $\hat{f}$ when the polynomial degree $(n_1^{(f)}, n_2^{(f)}, n_3^{(f)})$ is sufficiently large and when $\mathcal{T}_f$ is well approximated by the underlying Tucker approximation $\hat{\mathcal{T}}$. Neither of these properties can be guaranteed in Phases 1 and 2 alone. Therefore in a final step, Chebfun3 verifies the accuracy by comparing f and the computed approximation at Halton points [239]. If the estimated error is too large, the whole algorithm is restarted on a finer coarse grid from Phase 1.

3.1.4 Disadvantages

The Chebfun3 algorithm often requires unnecessarily many function evaluations. As we illustrate in the following, this is due to redundancy among the mode-2 and mode-3 fibers. For this purpose we collect all (refined) mode-2 fibers $S_i(:,L_i)$ in the block term decomposition (3.3) into the columns of a big matrix $V_m^{\mathsf{BTD}} = \begin{bmatrix} S_1(:,L_1) & \cdots & S_m(:,L_m) \end{bmatrix}$, where m is the number of steps of the outer ACA (3.1). As will be demonstrated with an example below, the matrix V_m^{BTD} is often observed to have low numerical rank, which in turn allows to represent its column space by much fewer columns, that is, much fewer mode-2 fibers. As the accuracy of the column space determines the accuracy of the Tucker decomposition after the compression, this implies that the other mode-2 fibers in V_m^{BTD} are redundant.

Let us now consider the block term decomposition[*] (3.3) for the function

$$f(x,y,z) = \frac{1}{1 + 25\sqrt{x^2 + y^2 + z^2}}.$$

In Figure 3.3 the numerical rank and the number of columns of V_m^{BTD} are compared. For $m = 10$ the approximation of the slices S_{i_1} to $S_{i_{10}}$ leads to a total of 153 mode-2 fibers,

---

[*] Note that the accuracy verification in Phase 3 fails once for this function. Here we only consider to block term decomposition obtained after restarting the procedure.

the sum of the corresponding red and blue bars in Figure 3.3. In contrast, their numerical rank (blue bar) is only 19. Thus, the red bar can be interpreted as number of redundant mode-2 fibers. This happens since nearby slices tend to be similar. The total block term decomposition contains 18 slices and is compressed into a Tucker decomposition with multilinear rank $(17, 19, 19)$. It contains 242 redundant fibers, the refinement requires 192 function evaluations for each of them. Note that the asymmetry in the rank of the Tucker decomposition is caused by the asymmetry of the block term decomposition.

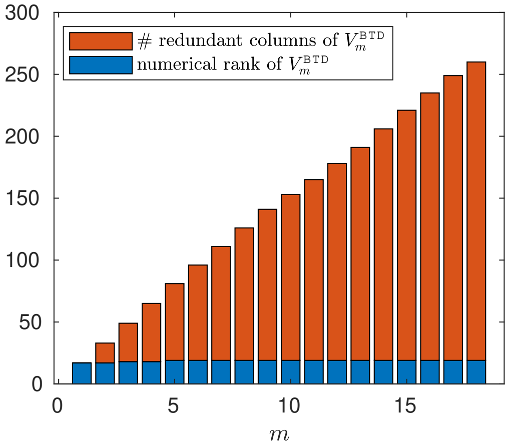


Figure 3.3 – Numerical rank and number of redundant columns (= total number of columns - numerical rank) of the matrix V_m^{BTD}, whose columns are given by the refined version of the mode-2 fibers determined after m steps of the outer ACA (3.1) in Chebfun3.

Another disadvantage is that Chebfun3 always requires the full evaluation of $\mathcal{T}_c$ in Phase 1. This becomes expensive when a large size $(n_1^{(c)} + 1) \times (n_2^{(c)} + 1) \times (n_3^{(c)} + 1)$ is needed in order to properly identify suitable fibers.

3.2 Novel algorithm: Chebfun3F

In this section, we describe our novel algorithm Chebfun3F to compute an approximation of the form (2.11). The goal of Chebfun3F is to the avoid the redundant function evaluations observed in Chebfun3. While the structure of Chebfun3F is similar to Chebfun3, consisting of 3 phases to identify/refine fibers and compute a Tucker decomposition, there is a major difference in Phase 1. Instead of proceeding via slices, we directly identify mode-ℓ fibers of $\mathcal{T}_c$ for building factor matrices. The core tensor is constructed in Phase 3.

3.2.1 Phase 1: Fiber indices and factor matrices

As in Chebfun3, the coarse tensor $\mathcal{T}_c \in \mathbb{R}^{(n_1^{(c)}+1)\times(n_2^{(c)}+1)\times(n_3^{(c)}+1)}$ is initially defined as evaluation tensor (2.18) with $(n_1, n_2, n_3) = (16, 16, 16)$. We seek to compute full rank factor matrices $U_c \in \mathbb{R}^{(n_1^{(c)}+1)\times r_1}$, $V_c \in \mathbb{R}^{(n_2^{(c)}+1)\times r_2}$ and $W_c \in \mathbb{R}^{(n_3^{(c)}+1)\times r_3}$ such that the orthogonal projection of $\mathcal{T}_c$ onto the span of the factor matrices is an accurate approximation of $\mathcal{T}_c$, i.e.

$$\mathcal{T}_c \approx \mathcal{T}_c \times_1 U_c(U_c^T U_c)^{-1}U_c^T \times_2 V_c(V_c^T V_c)^{-1}V_c^T \times_3 W_c(W_c^T W_c)^{-1}W_c^T. \tag{3.4}$$

Additionally, we require that the columns in U_c, V_c, W_c contain fibers of $\mathcal{T}_c$.

In the existing literature, algorithms to compute such factor matrices include the Higher Order Interpolatory Decomposition [277], which is based on a rank revealing QR decomposition, and the Fiber Sampling Tensor Decomposition [54], which is a generalization of the CUR decomposition. We propose a novel algorithm, which in contrast to the existing algorithms does not require the evaluation of the full tensor $\mathcal{T}_c$. We follow the ideas of TT-cross [251, 281] and its variants such as the Schur-Cross3D [271] and the ALS-cross [101]. After the publication of [100] several algorithms for this purpose have been suggested in [3]. The Tucker format with factor matrices containing fibers has been called generalized tensor CUR decomposition in [157], mode-wise tensor decomposition in [53] and tensor fiber CUR decomposition in [52].

Initially, we randomly choose index sets $\tilde{I}, \tilde{J}, \tilde{K}$ by partitioning $\{0, \dots, 16\}$ into 6 subsets and sampling one index from each subset for each index set. In the first step, we apply Algorithm 1 to $(\mathcal{T}_c(:, \tilde{J}, \tilde{K}))^{\{1\}}$. Note that this needs only $36 \cdot (n_1^{(c)} + 1)$ evaluations of the function f, in contrast to $(n_1^{(c)} + 1)(n_2^{(c)} + 1)(n_3^{(c)} + 1)$ values in the whole tensor $\mathcal{T}_c$. The selected r_1 columns serve as a first candidate for the factor matrix U_c. The index set $\tilde{I}$ is set to the row indices selected by Algorithm 1 (see Figure 3.4). We use the updated index set and apply Algorithm 1 to $(\mathcal{T}_c(\tilde{I}, : \tilde{K}))^{\{2\}}$ analogously, which yields V_c and an updated $\tilde{J}$. From $(\mathcal{T}_c(\tilde{I}, \tilde{J}, :))^{\{3\}}$ we obtain W_c and $\tilde{K}$. We repeat this process in an alternating fashion with the updated index sets, which leads to potentially improved factor matrices. Following the ideas of Chebfun3, we check after each iteration whether the ratios $r_1/(n_1^{(c)} + 1)$, $r_2/(n_2^{(c)} + 1)$ and $r_3/(n_3^{(c)} + 1)$ surpass the heuristic threshold $(2\sqrt{2})^{-1}$. If this is the case, we increase the size of the coarse tensor $n_\ell^{(c)}$ to $\left\lfloor \sqrt{2}^{\left\lfloor 2\log_2(n_\ell^{(c)}+1)+1 \right\rfloor} \right\rfloor$ and restart the whole process by reinitializing $\tilde{I}, \tilde{J}, \tilde{K}$ with r_1, r_2, r_3 random indices respectively.

It is not clear a priori how many iterations are needed to attain an approximation (3.4) that yields a Tucker approximation (2.11) which passes the accuracy verification in Phase 3. In numerical experiments, it has usually proven to be sufficient to stop after the second iteration, during which the coarse grid has not been refined, or when $|\tilde{I}| \leq 1$, $|\tilde{J}| \leq 1$ or $|\tilde{K}| \leq 1$. This is formalized in Algorithm 4. Note that U_c, V_c, W_c are full rank by construction, since Algorithm 1 stops based on the tolerance $\varepsilon \geq 0$. In many cases, we found that the numbers of columns in the factor matrices are equal to the multilinear rank of the truncated HOSVD [88] of $\mathcal{T}_c$ with the same tolerance.

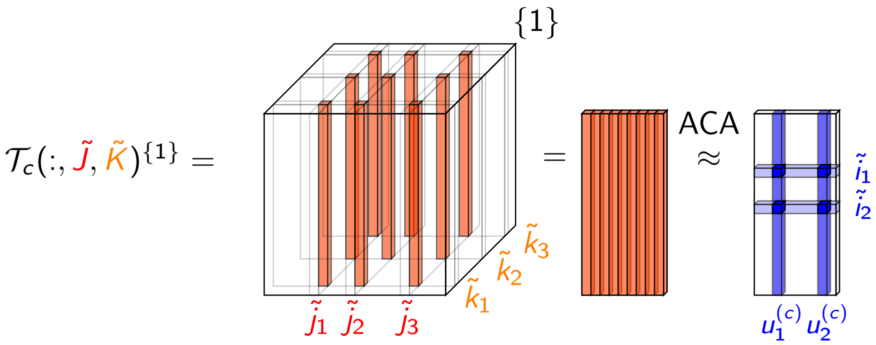


Figure 3.4 – Visualization of applying ACA (Algorithm 1) to a matricization of a subtensor.

Algorithm 4 Factor Matrix Computation

```
1: **Input:** $f$, $(n_1^{(c)}, n_2^{(c)}, n_3^{(c)})$, $(r_1, r_2, r_3)$
2: **Output:** $U_c, V_c, W_c$, $(n_1^{(c)}, n_2^{(c)}, n_3^{(c)})$, $(r_1, r_2, r_3)$
3: Let $\mathcal{T}_c(i, j, k) = f(x_i^{(1)}, x_j^{(2)}, x_k^{(3)})$ be a function that evaluates the values of $f$ on a $(n_1^{(c)} + 1) \times (n_2^{(c)} + 1) \times (n_3^{(c)} + 1)$ Chebyshev grid on demand.
4: initialize $\tilde{J}, \tilde{K}$ with $r_2, r_3$ randomly chosen indices in $\left\{0, \ldots, n_1^{(c)}\right\}, \left\{0, \ldots, n_2^{(c)}\right\}$
5: **for** iterations = 1:2
6:     compute ACA of $\mathcal{T}_c(:, \tilde{J}, \tilde{K})^{\{1\}} \to U_c \in \mathbb{R}^{(n_1^{(c)}+1) \times r_1}$ = selected columns, $\tilde{I}$ = selected row indices
7:     compute ACA of $\mathcal{T}_c(\tilde{I}, :, \tilde{K})^{\{2\}} \to V_c \in \mathbb{R}^{(n_2^{(c)}+1) \times r_2}$ = selected columns, $\tilde{J}$ = selected row indices
8:     compute ACA of $\mathcal{T}_c(\tilde{I}, \tilde{J}, :)^{\{3\}} \to W_c \in \mathbb{R}^{(n_3^{(c)}+1) \times r_3}$ = selected columns, $\tilde{K}$ = selected row indices
9:     **if** the multilinear ranks get too large $\to$ adjust $(n_1^{(c)}, n_2^{(c)}, n_3^{(c)})$ and go to line 3
10:    **if** $r_1 \leq 1$ or $r_2 \leq 1$ or $r_3 \leq 1 \to$ return
```

3.2.2 Phase 2: Refinement of the factors

In Phase 2, the fibers in U_c, V_c, W_c are refined using the Chebfun heuristic [16] as in Chebfun3 (see Section 3.1.2). This leads to full rank factor matrices $U_f \in \mathbb{R}^{(n_1^{(f)}+1) \times r_1}$, $V_f \in \mathbb{R}^{(n_2^{(f)}+1) \times r_2}$ and $W_f \in \mathbb{R}^{(n_3^{(f)}+1) \times r_3}$ containing the refined fibers of $\mathcal{T}_f$, where $\mathcal{T}_f$ denotes the evaluation tensor (2.18) of size $(n_1^{(f)} + 1) \times (n_2^{(f)} + 1) \times (n_3^{(f)} + 1)$. This phase needs only $\mathcal{O}(\sum_{\ell=1}^{3} (n_\ell^{(f)} + 1) r_\ell)$ evaluations of f.

3.2.3 Phase 3: Reconstruction of the core tensor

In the final Phase of Chebfun3F, we compute the core tensor $\hat{\mathcal{C}}$ to obtain the approximation $\mathcal{T}_f \approx \hat{\mathcal{C}} \times_1 U_f \times_2 V_f \times_3 W_f$.

In principle, the best approximation (with respect to the Frobenius norm) for fixed factor matrices U_f, V_f, W_f is obtained by orthogonal projections as in (2.7) by

$$\mathcal{T}_f \approx \mathcal{T}_f \times_1 U_f(U_f^T U_f)^{-1}U_f^T \times_2 V_f(V_f^T V_f)^{-1}V_f^T \times_3 W_f(W_f^T W_f)^{-1}W_f^T.$$

Such an approach comes with the major disadvantage that the full evaluation of $\mathcal{T}_f$ is required. This can be circumvented by instead using oblique projections. Applying oblique projections in all three modes analogous to (2.3) yields

$$\begin{aligned}\mathcal{T}_f \approx \hat{\mathcal{T}} =& \mathcal{T}_f \times_1 U_f(\Phi_I^T U_f)^{-1}\Phi_I^T \times_2 V_f(\Phi_J^T V_f)^{-1}\Phi_J^T \times_3 W_f(\Phi_K^T W_f)^{-1}\Phi_K^T \\ =& \underbrace{\left(\underbrace{(\mathcal{T}_f \times_1 \Phi_I^T \times_2 \Phi_J^T \times_3 \Phi_K^T)}_{=\mathcal{T}_f(I,J,K)} \times_1 (\Phi_I^T U_f)^{-1} \times_2 (\Phi_J^T V_f)^{-1} \times_3 (\Phi_K^T W_f)^{-1}\right)}_{=\hat{\mathcal{C}}} \\ & \times_1 U_f \times_2 V_f \times_3 W_f,\end{aligned}$$

for index sets I, J, K and matrices Φ_I, Φ_J, Φ_K defined analogously as in (2.3). Note that the computation of the $\mathcal{T}_f$ only requires $r_1 \cdot r_2 \cdot r_3$ additional evaluations of f. To avoid the potentially ill-conditioned matrices $(\Phi_I^T U_f)^{-1}$, $(\Phi_J^T V_f)^{-1}$,$(\Phi_K^T W_f)^{-1}$, we compute the orthogonal matrices Q_U, Q_V, Q_W in the QR decompositions of U_f, V_f, W_f. Note that $U_f(\Phi_I^T U_f)^{-1} = Q_U(\Phi_I^T Q_U)^{-1}$ and $(\mathcal{T}_f(I,J,K) \times_1 (\Phi_I^T U_f)^{-1}) \times_1 U_f = \mathcal{T}_f(I,J,K) \times_1 U_f(\Phi_I^T U_f)^{-1}$. To determine the index sets I, J, K we apply DEIM [65], presented in Algorithm 2, to the matrices Q_U, Q_V, Q_W. This leads to the Tucker decomposition used in Chebfun3F

$$\begin{aligned}\hat{\mathcal{T}} &= \mathcal{T}_f \times_1 Q_U(\Phi_I^T Q_U)^{-1}\Phi_I^T \times_2 Q_V(\Phi_J^T Q_V)^{-1}\Phi_J^T \times_3 Q_W(\Phi_K^T Q_W)^{-1}\Phi_K^T \\ &= \mathcal{T}_f(I,J,K) \times_1 Q_U(\Phi_I^T Q_U)^{-1} \times_2 Q_V(\Phi_J^T Q_V)^{-1} \times_3 Q_W(\Phi_K^T Q_W)^{-1}. \qquad (3.5)\end{aligned}$$

Note that in general neither Q_U nor $Q_U(\Phi_I^T Q_U)^{-1}$ contain fibers of $\mathcal{T}_f$. This issue will be discussed in Section 3.4.

3.2.4 Chebfun3F algorithm

Having computed the Tucker factors U_f, V_f and W_f in Phase 2, and the core $\hat{\mathcal{C}}$ in Phase 3, we obtain the approximation (2.24) of the form (2.11). Following Chebfun3, we compute the Chebyshev interpolants of the columns of U_f, V_f, W_f using Chebfun [27]. We also perform an accuracy verification for $\hat{f}$ by comparing its evaluations at Halton points to the original f. If the difference of the evaluations is too large, we restart the whole algorithm up to ten times using a finer coarse grid. Additionally, we modify the ranks such that if $r_{\ell_1} \leq 2$ we set $r_{\ell_2} = \max(6, 2r_{\ell_2})$ and $r_{\ell_1} = 3$ for $\ell_1 \neq \ell_2$, and after the forth restart we set $r_\ell = 2r_\ell$ for $\ell = 1, 2, 3$. This ensures that the multilinear ranks can grow in Phase 1. The overall Chebfun3F algorithm is formalized in Algorithm 5.

Algorithm 5 Chebfun3F

```
Input: function $f(x,y,z)$, a procedure $\mathcal{T}_f(i,j,k) = f(x_i^{(1)}, x_j^{(2)}, x_k^{(3)})$ that evaluates the values of $f$ on a $(n_1^{(f)}+1) \times (n_2^{(f)}+1) \times (n_3^{(f)}+1)$ Chebyshev grid on demand, a stopping tolerance $\varepsilon > 0$.
Output: approximation $\hat{f}(x,y,z) = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} \sum_{k=1}^{r_3} \mathcal{C}_{ijk} u_i(x) v_j(y) w_k(z)$
Initialization: $(n_1^{(c)}, n_c^{(c)}, n_3^{(c)}) = (16,16,16)$, $(r_1,r_2,r_3) = (6,6,6)$
Phase 1:
    apply Algorithm 4 to compute the factor matrices $U_c, V_c, W_c$ and to update $(n_1^{(c)}, n_2^{(c)}, n_3^{(c)})$, $(r_1,r_2,r_3)$
Phase 2:
    $(n_1^{(f)}, n_2^{(f)}, n_3^{(f)}) = (n_1^{(c)}, n_2^{(c)}, n_3^{(c)})$,    $U_f = U_c$, $V_f = V_c$, $W_f = W_c$
    while Chebfun heuristic in [16] to decide if $U_f$ contains a sufficient number of entries is not satisfied
        $n_1^{(f)} + 1 = 2n_1^{(f)}$,    refine $U_f$ such that its columns have length $n_1^{(f)}$
    proceed analogously to obtain $V_f, W_f$
Phase 3:
    $[Q_U, \sim] = \mathrm{qr}(U_f)$,    $I = \mathrm{DEIM}(Q_U)$,    $U = Q_U \cdot Q_U(I,:)^{-1}$
    $[Q_V, \sim] = \mathrm{qr}(V_f)$,    $J = \mathrm{DEIM}(Q_V)$,    $V = Q_V \cdot Q_V(J,:)^{-1}$
    $[Q_W, \sim] = \mathrm{qr}(W_f)$,    $K = \mathrm{DEIM}(Q_W)$,    $W = Q_W \cdot Q_W(K,:)^{-1}$
    compute the Chebyshev interpolants $u_i(x), v_j(y), w_k(z)$ based on $U, V, W$ (see Section 2.3.4)
    $\mathcal{C} = \mathcal{T}_f(I,J,K)$,   $\hat{f}(x,y,z) = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} \sum_{k=1}^{r_3} \mathcal{C}_{ijk} u_i(x) v_j(y) w_k(z)$
    if $|f(x,y,z) - \hat{f}(x,y,z)| > 10\varepsilon$ at Halton points $(x,y,z) \in [-1,1]^3$
        modify the ranks $r_\ell$ if they are too small
        restart from Phase 1 with $n_\ell^{(c)} = \left\lfloor \sqrt{2}^{\left\lfloor 2\log_2(n_\ell^{(c)}+1)+1 \right\rfloor} \right\rfloor$
```

Remark 3.1. *In Chebfun3 upper bounds for the multilinear rank, polynomial degree and grid sizes are prescribed. The tolerances in the accuracy verification and in the ACA are initially set close to machine precision or provided by the user. Tolerance issues are avoided by relaxing these tolerances adaptively based on the computed function evaluations. In Chebfun3F, we handle these technicalities in the same manner.*

3.2.5 Existence of a quasi-optimal Chebfun3F approximation

Due to the many heuristic ingredients in the Chebfun3F algorithm, it is difficult to analyze the convergence of the whole algorithm. Instead, we discuss the existence and error analysis of a specific Chebfun3F reconstruction. Lemma 2.2 shows how we can bound the approximation error depending on $\|\mathcal{T}-\hat{\mathcal{T}}\|_\infty$. Theorem 3.1 provides a bound for this error for a tensor approximation of the format (3.5) with specifically chosen fibers and index sets. The best approximation in the format will be at least as good. Although Chebfun3F is not guaranteed to return these specific fibers and index sets, it is hoped that its error is not much larger.

Theorem 3.1. *Consider* $\mathcal{T} \in \mathbb{R}^{n_1\times n_2\times n_3}$ *of multilinear rank at least* (r_1, r_2, r_3). *Let* Q_U, Q_V, Q_W *denote orthonormal bases of* r_1, r_2, r_3 *selected mode-*$1, 2, 3$ *fibers of* $\mathcal{T}$ *respectively. Given index sets* I, J, K *we consider a Tucker decomposition of the form*

$$\hat{\mathcal{T}} = \mathcal{T} \times_1 Q_U(\Phi_I^T Q_U)^{-1}\Phi_I^T \times_2 Q_V(\Phi_J^T Q_V)^{-1}\Phi_J^T \times_3 Q_W(\Phi_K^T Q_W)^{-1}\Phi_K^T.$$

There exists a choice of fibers and indices such that

$$\begin{aligned}\|\mathcal{T}-\hat{\mathcal{T}}\|_\infty \le \Big(&\sqrt{q(r_1,n_1)\cdot(r_1+1)} + \sqrt{q(r_1,n_1)\cdot q(r_2,n_2)\cdot(r_2+1)} \\ &+ \sqrt{q(r_1,n_1)\cdot q(r_2,n_2)\cdot q(r_3,n_3)\cdot(r_3+1)}\Big)\, \|\mathcal{T}-\hat{\mathcal{T}}_{\mathsf{best}}\|_F,\end{aligned}$$

where $q(r,n) = \sqrt{1+r(n-r)}$, *and* $\hat{\mathcal{T}}_{\mathsf{best}}$ *is the best Tucker approximation of* $\mathcal{T}$ *with multilinear rank at most* (r_1, r_2, r_3).

Proof. Using Frobenius norm properties and Lemma 2.1, we obtain

$$\begin{aligned}\|\mathcal{T}-\hat{\mathcal{T}}\|_\infty \le &\|\mathcal{T}-\hat{\mathcal{T}}\|_F \le \|(\Phi_I^T Q_U)^{-1}\|_2\|(I-Q_UQ_U^T)\mathcal{T}^{\{1\}}\|_F \\ &+ \|(\Phi_I^T Q_U)^{-1}\|_2\|(\Phi_J^T Q_V)^{-1}\|_2\|(I-Q_VQ_V^T)\mathcal{T}^{\{2\}}\|_F \\ &+ \|(\Phi_I^T Q_U)^{-1}\|_2\|(\Phi_J^T Q_V)^{-1}\|_2\|(\Phi_K^T Q_W)^{-1}\|_2\|(I-Q_WQ_W^T)\mathcal{T}^{\{3\}}\|_F.\end{aligned} \tag{3.6}$$

From [138, Lemma 2.1] it follows that there exists an index set I such that

$$\|(\Phi_I^T Q_U)^{-1}\|_2 \le \sqrt{1+r_1(n_1-r_1)}. \tag{3.7}$$

From [93, Theorem 8] with the role of rows and columns interchanged it follows that we can select mode-1 fibers U of $\mathcal{T}$ such that

$$\|(I - Q_U Q_U^T)\mathcal{T}^{\{1\}}\|_F = \|(I - U(U^TU)^{-1}U^T)\mathcal{T}^{\{1\}}\|_F \leq \sqrt{r_1 + 1}\|\mathcal{T} - \hat{\mathcal{T}}_{\mathsf{best}}\|_F. \quad (3.8)$$

Analogous bounds hold for $\|(\Phi_J^T Q_V)^{-1}\|_2$, $\|(\Phi_K^T Q_W)^{-1}\|_2$, $\|(I - Q_V Q_V^T)\mathcal{T}^{\{2\}}\|_F$ and $\|(I - Q_W Q_W^T)\mathcal{T}^{\{3\}}\|_F$. Applying the bounds (3.7) and (3.8) to the factors in (3.6) yields the claimed result. □

Remark 3.2. *If one uses orthogonal instead of oblique projections in Phase 3, Corollary 6 in [79] yields a bound similar to Theorem 3.1. Whilst the index sets obtained from DEIM [66] yield small errors in practice, their theoretical upper bounds for* $\|(\Phi_I^T Q_U)^{-1}\|_2$ *grow exponentially in* r*. In contrast, the strong rank-revealing QR decomposition [149] yields an index set for which a bound similar to inequality* (3.7) *is known [105, Lemma 2.1].*

Remark 3.3. *Note that the bound in Theorem 3.1 is the worst case bound for the optimal choice of index sets. This does not present the whole picture as even suboptimal index sets might yield a much better approximation in practice. To quantify the quality of the approximation in practice, we computed Chebfun3F approximations for the functions* $f(x, y, z) = \log(1 + x^2 + y^2 + z^2)$, $f(x, y, z) = 1/(1 + x^2 + y^2 + z^2)$, $f(x, y, z) = \exp(xyz)$ *and compared* $\|\mathcal{T}_f - \hat{\mathcal{T}}\|_\infty$ *and* $\|\mathcal{T}_f - \mathcal{T}_{\text{HOSVD}}\|_\infty$*, where* $\hat{\mathcal{T}}$ *is computed using Chebfun3F and* $\mathcal{T}_{\text{HOSVD}}$ *denotes the truncated HOSVD of* $\mathcal{T}_f$ *with multilinear ranks equal to those of* $\hat{\mathcal{T}}$*. We observed that both errors differ by at most a factor of* 2*. Even though the truncated HOSVD does not focus on* $\|\cdot\|_\infty$*, it still can serve as a good proxy. Hence by Lemma 2.2, the error in Chebfun3F is comparable to the error obtained from the truncated HOSVD.*

3.2.6 Comparison of the theoretical cost

Assume f can be approximated accurately in Tucker format (2.11) with multilinear rank (r, r, r) and polynomial degrees (n, n, n), $n \geq r$. In a highly idealized setting Chebfun3 and Chebfun3F refine the coarse grid in Phase 1 until $(n_\ell^{(c)} + 1) > (2\sqrt{2})r$ and identify fibers on this coarse grid. These fibers are refined until $n_\ell^{(f)} \geq n$ and lead to Tucker approximations which pass the accuracy check in Phase 3. Under these circumstances, both Chebfun3 and Chebfun3F use $\mathcal{O}(r^3)$ function evaluations in Phase 1 (see Section 2.2 in [163]). In total, Chebfun3 requires $\mathcal{O}(nr^2)$ function evaluations [163, Proposition 2.1]), whereas Chebfun3F only requires $\mathcal{O}(r^3 + nr)$, since fewer fibers are refined in Phase 2. We want to emphasize that, in general, it is not guaranteed that this $n_\ell^{(c)}$ suffices to identify fibers leading to an accurate approximation. We summarize the breakdown of anticipated costs in Table 3.1.

	Chebfun3	Chebfun3F
Phase 1	$\mathcal{O}(r^3)$	$\mathcal{O}(r^3)$
Phase 2	$\mathcal{O}(nr^2)$	$\mathcal{O}(nr)$
Phase 3	0	$\mathcal{O}(r^3)$
Total	$\mathcal{O}(r^3+nr^2)$	$\mathcal{O}(r^3+nr)$

Table 3.1 – Anticipated numbers of functions evaluations in Chebfun3 and Chebfun3F.

3.3 Numerical results

In this section, we present numerical experiments[†] to compare Chebfun3F and Chebfun3. The main focus lies on the number of function evaluations required to compute the approximation in Tucker format (2.11). Unless mentioned otherwise the tolerance for the ACA and the accuracy check are initially set close to machine precision.

3.3.1 Chebfun3 vs. Chebfun3F

In Section 3.1.4 we illustrated that the function

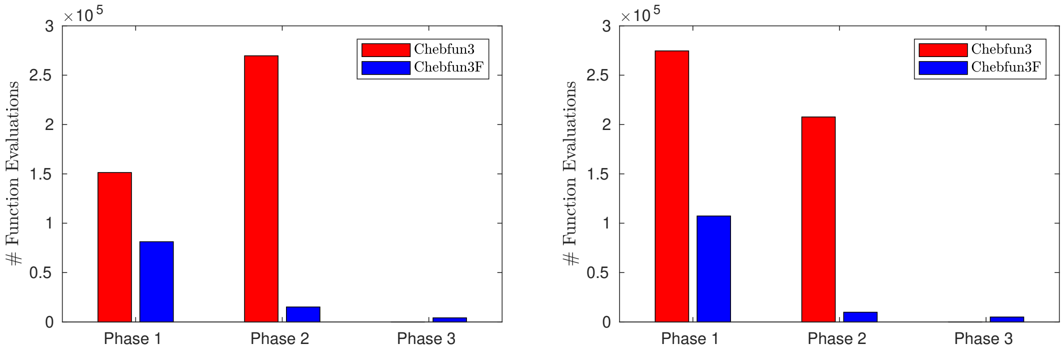
$$f(x,y,z)=\frac{1}{1+25\sqrt{x^2+y^2+z^2}},$$

leads to a lot of redundant fibers in Chebfun3. For this function, for both Chebfun3 and Chebfun3F, the accuracy check at the end of Phase 3 fails and the computation is restarted on a finer coarse grid, which leads to an approximation that passes the test. Overall 903 364 function evaluations are required by Chebfun3, 421 041 of them are used before the restart. In comparison, Chebfun3F only needs 222 546 function evaluation in total and 100 496 before the restart. In the final accuracy check, the estimated error for Chebfun3 is $7.8\cdot 10^{-13}$ and $3.6\cdot 10^{-13}$ for Chebfun3F, i.e. both approximations achieve around the same accuracy.

In Figure 3.5, the function evaluations per phase are juxtaposed. The figure shows, that Chebfun3 requires a large number of function evaluations in Phase 2. This is caused by the redundant fibers. In Phase 1, Chebfun3F requires fewer function evaluations, since $\mathcal{T}_c$ is not evaluated completely. Whilst Chebfun3 only requires function evaluations in Phase 3 for the accuracy check, Chebfun3F additionally needs to compute the core tensor, which only leads to a small number of function evaluations compared to the other phases.

Remark 3.4. *The number of function evaluations required by Chebfun3F depends on the random initialization of the index sets $\tilde{I},\tilde{J},\tilde{K}$ in Algorithm 4. We computed the Chebfun3F approximation of f for 1 000 different random initializations and observed*

[†] The MATLAB code to reproduce these results is available from https://github.com/cstroessner/Chebfun3F.



(a) Before Restarting (b) After Restarting

Figure 3.5 – Comparison of the function evaluations used by Chebfun3 and Chebfun3F to approximate $f(x,y,z) = (1 + 25\sqrt{x^2+y^2+z^2})^{-1}$. Both algorithms are restarted due to failing the accuracy check once. Evaluations before the restart are depicted in (a), evaluations after the restart in (b). The evaluations are subdivided into phases corresponding to the phases in Section 3.1 and Section 3.2 respectively.

numbers of function evaluations ranging from 213 391 *to* 226 073 *with mean* 221 802.6 *and variance* $4.96 \cdot 10^6$. *Similarly mild fluctuations have been observed for all other functions tested.*

In Figure 3.6, the required function evaluations are depicted for four different functions. The corresponding computing times are depicted in Table 3.2. Again both algorithms lead to approximations of similar accuracy and Chebfun3F requires fewer function evaluations than Chebfun3. For

$$f_1(x,y,z) = \exp(-\sqrt{(x-1)^2 + (y-1)^2 + (z-1)^2}) \tag{3.9}$$

Chebfun3 requires the refinement of a huge number of redundant fibers in Phase 2. The evaluations for

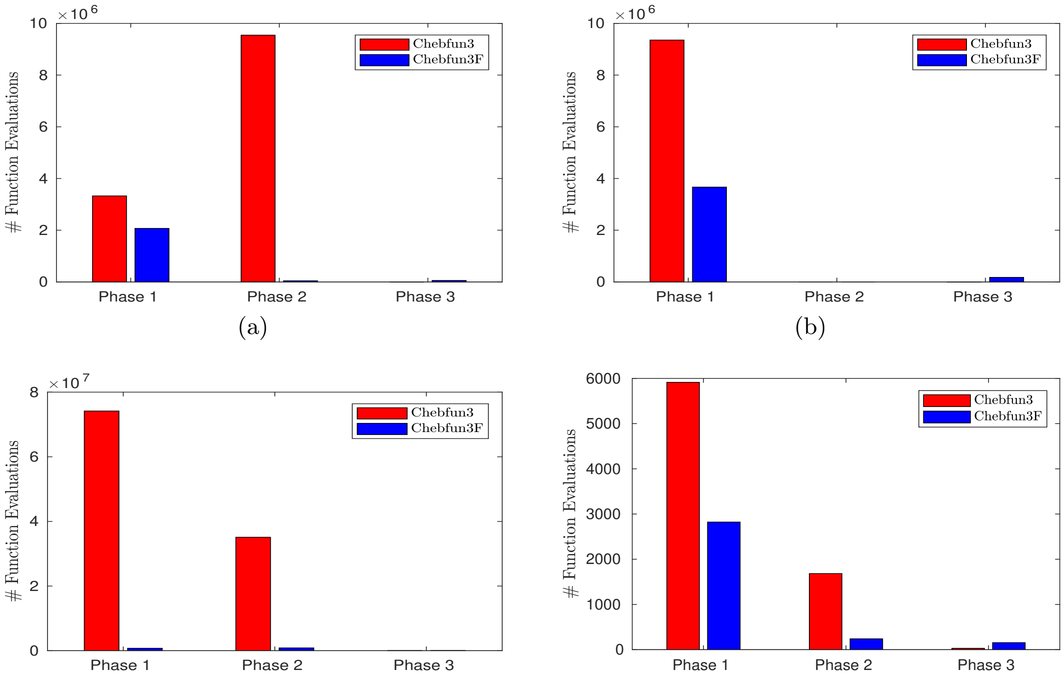
$$f_2(x,y,z) = [\cosh(3(x+y+z))]^{-2} \tag{3.10}$$

differ most in Phase 1, where Chebfun3F benefits from not evaluating $\mathcal{T}_c$ completely. No additional refinement is required in Phase 2. For the function

$$f_3(x,y,z) = \frac{10^5}{1 + 10^5(x^2+y^2+z^2)}. \tag{3.11}$$

Chebfun3F reduces the number of required function evaluations by more than 98% to 1 603 693 from 109 269 332 required by Chebfun3.

In certain cases Chebfun3 outperforms Chebfun3F. This can happen in degenerated situations. For instance, the function $f(x,y,z) = \tanh(5(x+z))\exp(y)$ requires a Tucker



(a) (b)

(c) (d)

Figure 3.6 – Total number of function evaluations per Phase in Chebfun3 and Chebfun3F for the functions: (a) f_1 (b) f_2, (c) f_3 as defined in (3.9)-(3.11) (d) the parametric PDE model (3.12)-(3.13) with prescribed tolerance 10^{-9}.

	f_1	f_2	f_3	PDE
Chebfun3	3.44	3.20	15.80	408.62
Chebfun3F	0.30	0.77	0.56	100.19

Table 3.2 – Computing times in MATLAB in seconds for Chebfun3 and Chebfun3F approximations of f_1, f_2, f_3 as defined in (3.9)-(3.11) and of the parametric PDE model (3.12)-(3.13) with prescribed tolerance 10^{-9}.

decomposition with rank $r = [71, 1, 71]$. In this case, Chebfun3F heavily relies on the heuristic to increase (r_1, r_2, r_3) when restarting and requires 1 641 712 function evaluations compared to 1 128 061 in Chebfun3. Other functions that are difficult to approximate with Chebfun3F include (numerically) locally supported functions, such as a trivariate normal distribution with very small entries in the covariance matrix, for which identifying non-zero fibers in Phase 1 without fully evaluating $\mathcal{T}_c$ might be difficult.

Application: Uncertainty quantification

Algorithms in uncertainty quantification, such as the Metropolis-Hastings method, often require the repeated evaluations of a parameter depended quantity of interest [307]. In many applications, the evaluation of this quantity requires the solution of a PDE depending on the parameters. To speed up computations, the mapping from the parameters to the quantity of interest is often replaced by a surrogate model [341]. In the context of models with three parameters (or after a dimension reduction to three parameters [77]), Chebfun3/Chebfun3F could be a suitable surrogate.

We consider the parametric elliptic PDE model problem on $\Omega = [1,3]^2$

$$\nabla\cdot((p_1 f_1(x,y) + p_2 f_2(x,y) + p_3 f_3(x,y))\nabla u(x,y)) = 1 \qquad (x,y) \in \Omega, \tag{3.12}$$

$$u(x,y) = 0 \qquad \in \partial\Omega, \tag{3.13}$$

with parameters $(p_1, p_2, p_3) \in [1,3]^3$ and functions $f_1(x,y) = \cos(x) + \sin(y) + 2$, $f_2(x,y) = \sin(x) + \cos(y) + 2$, and $f_3(x,y) = \cos(x^2 + y^2) + 2$. The quantity of interest is defined as point evaluation $u(1.5, 1.5)$. With Chebfun3F, we only need 3 217 PDE solves to compute an approximation with prescribed accuracy 10^{-9}, whereas Chebfun3 requires 7 626 as depicted in Figure 3.6(d).

3.3.2 Comparison to sparse grids

Lastly, we study how efficient Chebfun3 approximations are compared to sparse grids [50]. Sparse grids are a method to interpolate functions by projecting them onto a particular space. This space is obtained by selecting the most beneficial elements from a hierarchical basis. In the following, we use dimension-adaptive sparse grids based on a Chebyshev-Gauss-Lobatto grid with polynomial basis functions as implemented in the sparse grid interpolation toolbox [195].

We compare how the approximation error decays compared to the number of function evaluations. Therefore, we prescribe varying tolerances to the algorithms. In Figure 3.7 the error decay is plotted for sparse grids, Chebfun3 and Chebfun3F. In (a), the function already studied in section 3.1.4 is depicted. We observe that both Chebfun3F and Chebfun3 require fewer function evaluations than sparse grids to achieve the same accuracy. The sparse grids perform poorly, since the function is smooth, but the norms of the second mixed derivatives are rather large. In contrast, the sum of 10 Gaussians depicted in (b) is well suited for sparse grids. In this case sparse grids require fewer function evaluations than Chebfun3F and Chebfun3 for the same accuracy. In (c) the Chebfun3F and sparse grids perform about equally well for

$$f_4(x,y,z) = \log(x + yz + \exp(xyz) + \cos(\sin(\exp(xyz)))). \tag{3.14}$$

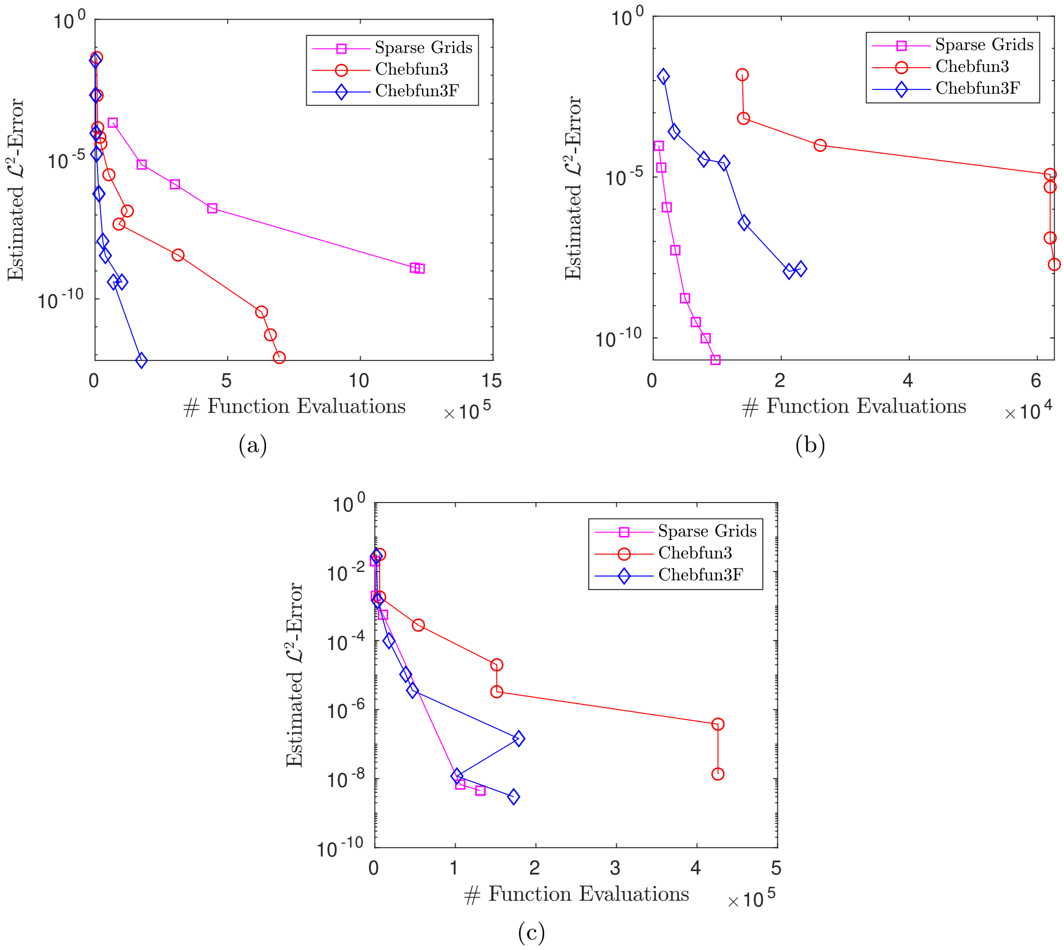


Figure 3.7 – Comparison of the number of function evaluations required to compute a Chebfun3, Chebfun3F and sparse grid approximation for the functions: (a) $f(x, y, z) = (1 + 25\sqrt{x^2 + y^2 + z^2})^{-1}$, (b) sum of 10 Gaussians, (c) f_4 as defined in (3.14). The algorithms are initialized with varying tolerances. Their $\mathcal{L}^2$ error is estimated at 1 000 sample points.

Note that the Chebfun3F heuristics only require a restart for a tolerance of around 10^{-7}. This leads to the outlier in the Chebfun3F graph. For an arbitrary, black-box function it is not clear a priory whether a sparse grid interpolation or a Tucker decomposition (2.11) is the more efficient type of approximation. Note that when the Tucker decomposition is the better approximation format, we can expect that Chebfun3F requires fewer function evaluations compared to Chebfun3.

We conclude that trivariate functions defined on tensor product domains can be approximated efficiently by combining tensorized Chebyshev interpolation and a low-rank Tucker approximation of the evaluation tensor. Our numerical experiments demonstrate that Chebfun3F typically requires fewer function evaluations to compute such an approximation of the same accuracy compared to Chebfun3.

3.4 Coefficient decay analysis

There is one major difference between the approximation computed using Chebfun3 and Chebfun3F as presented in Algorithm 5. The compression step in Phase 3 of Chebfun3 ensures that the Tucker approximation of the coefficient tensor (2.22) contains factor matrices with entries decaying to close to machine precision. This is not the case of Chebfun3F, where the corresponding factor matrices are of the form $F^{(1)}Q_U$ or $F^{(1)}Q_U(\Phi_I^T Q_U)^{-1}$ depending on if one treats the terms of the form $(\Phi_I^T Q_U)^{-1}$ in (3.5) as part of the factor matrix or as part of the core tensor. See Figure 3.8 for an example of this discrepancy.

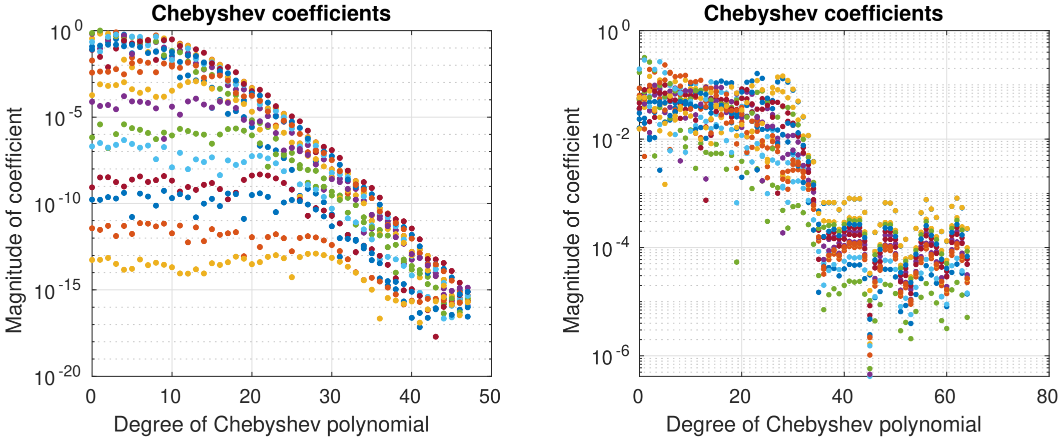


Figure 3.8 – We plot the decay of the univariate Chebyshev interpolation coefficients (2.13) of the functions $u_i^{(1)}$ in (2.24) for the approximations of $f(x, y, z) = \sin(\exp(x + y + z))$ computed using Chebfun3 (left) and Chebfun3F (right). The different colors correspond to different values of $i \in \{1, \ldots, r_1\}$.

Many functions to perform computations with univariate functions in the Chebfun package assume that the function is represented in terms of coefficients (2.13) that decay until reaching values close to machine precision. In Chebfun3F the coefficient tensor (2.19) computed from (3.5) has decaying coefficients when studied as a whole, but the coefficients stored in the factor matrices do not decay to machine precision when the core tensor is not taken into account. This implies that applying operations such as e.g. computing a derivative by a mode-ℓ multiplication with a differentiation matrix (see Section7.2) are working as expected when the whole coefficient tensor is taken into account. However, when we only consider the multiplication of the ℓth factor matrix with the differentiation matrix, heuristics [16] would state that the operation has not been performed accurately. Note that it might not be feasible to analyse the coefficient decay of the whole coefficient tensor in practice. In the following, we discuss how Chebfun3F approximations can be modified to circumvent this issue. Note that we can not simply replace $Q_U(\Phi_I^T Q_U)^{-1}$

in (3.5) by $U_f(\Phi_I^T U_f)^{-1}$, since this would lead to very ill-conditioned matrices $\Phi_I^T U_f$.

3.4.1 Observations in a two-dimensional setting

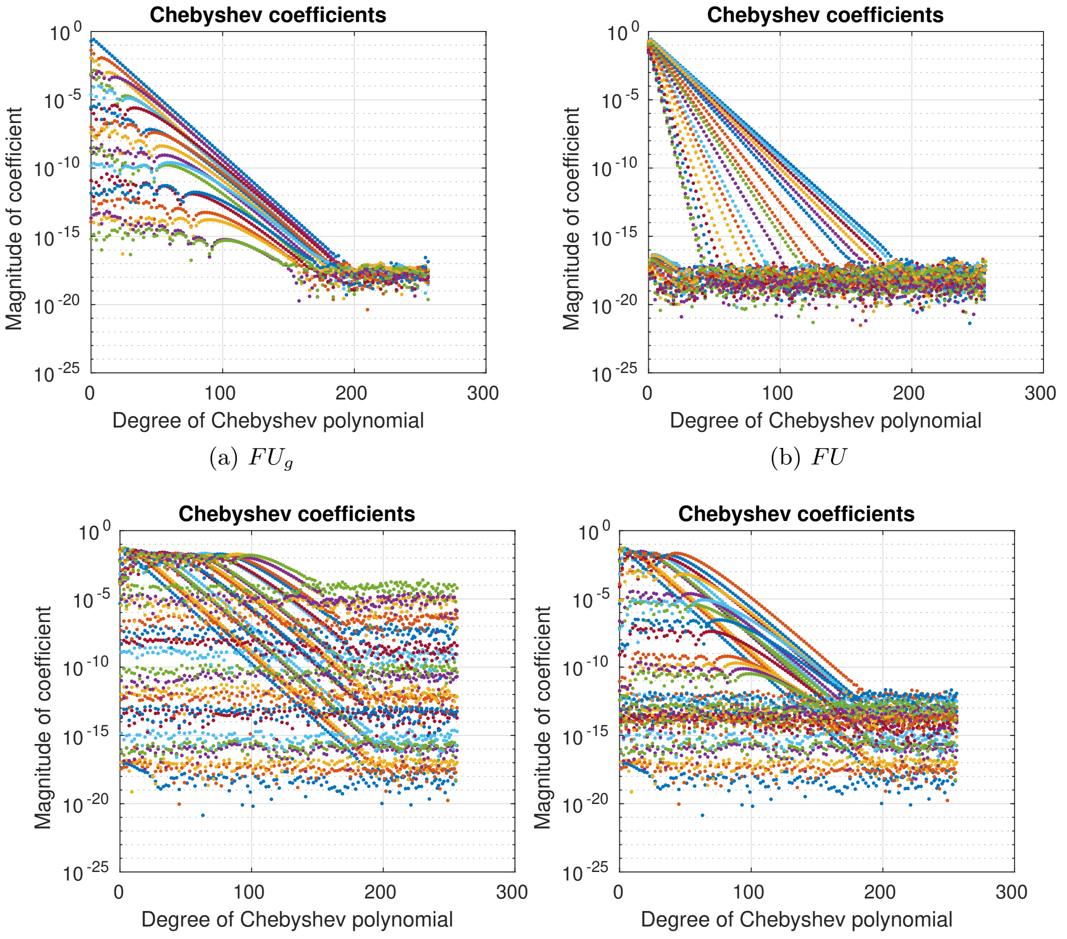
To study the aforementioned issue, we first analyze the problem in a two-dimensional setting using a fixed polynomial degree (n, n). Throughout this section, we denote the evaluation matrix (2.18) by T and the transformation matrix (2.14) by F. All numerical experiments are performed using the function $f(x, y) = 1/(1 + 25(x^2 + y^2))$ and $n = 257$.

Gaussian elimination In Chebfun2 [320], the matrix T is approximated using Gaussian elimination. The approximation is constructed iteratively using ACA. Instead of storing index sets I, J the approximation (2.1) is stored as sum of the rank-1 matrices subtracted in each iteration in line 7 of Algorithm 1. This can be written as $T \approx U_g C_g V_g^T$, where norms of the columns in U_g, V_g decay rapidly and the matrix $C_g \in \mathbb{R}^{r\times r}$ is a diagonal matrix which contains the inverses of the pivot elements determined in line 5. It is important to note that the columns in U_g are not directly fibers of the original function. The matrix C_g is very ill-conditioned, but the large entries in C_g cancel out with the small norms in the corresponding columns of U_g and V_g. The final approximation contains univariate interpolants (2.13) with coefficients given by the columns of FU_g, which decay as depicted in Figure 3.9(a).

Adaptive cross approximation If we were to apply ACA as presented in Algorithm 1 to T, we would obtain an approximation of the form $T \approx T(:, \tilde{J})T(\tilde{I}, \tilde{J})^{-1}T(\tilde{I}, :)$. Let $U = T(:, \tilde{J}), V = T(\tilde{I}, :)^T$. This leads to a final approximation containing Chebfuns with coefficients given by the columns of FU, which decay as depicted in Figure 3.9(b). The final approximation can be evaluated with an accuracy up to machine precision by using the backslash operator of MATLAB to avoid inverting $T(\tilde{I}, \tilde{J})^{-1}$. However, we have to point out that the inverse $T(\tilde{I}, \tilde{J})^{-1}$ (computed using MATLAB) is an ill-conditioned matrix with condition number around 10^{15}, which when used directly would lead to errors of order 10^0 in point evaluation of the approximation. In fact, there we could not find any matrix $C \in \mathbb{R}^{r\times r}$ such that $||T - UCV^T||_F^2 \leq 10^{-8}$ in double precision.

Note that neither Gaussian eliminations of the form $U_g C_g V_g^T$ nor the inversion using backslash can be generalized directly to the three-dimensional setting.

Two-dimensional Chebfun3F The two-dimensional equivalent of Chebfun3F would be to compute QR decompositions $U = Q_U R_U$, $V = Q_V R_R$ based on which we construct a decomposition of the form $T \approx Q_U \tilde{C} Q_V^T$, where $\tilde{C} \in \mathbb{R}^{r\times r}$. In contrast to the previous paragraph, where it was not possible to find a suitable matrix C, we can now find matrices $\tilde{C}$ that lead to accurate approximations of T. Using oblique projections based on DEIM (2.3) we obtain $\tilde{C} = Q_U(I,:)^{-1}T(I,J)Q_V(J,:)^{-T}$, where I, J are index sets obtained by applying DEIM to Q_U and Q_V. This is the only approach so far that can be generalized to higher dimensions, but the interpolation coefficients in the columns of the factor matrix FQ_U decay as depicted Figure 3.9(c) and would not be considered proper approximations of univariate functions in Chebfun.



(a) FU_g (b) FU

(c) FQ_U (d) $F\bar{Q}_U$

Figure 3.9 – Decay of the coefficients stored in the columns of FU_g, FU, FQ_U, $F\bar{Q}_U$ defined in Sections 3.4.1 and 3.4.2.

3.4.2 Potential solutions in the two-dimensional setting

From the previous section, we conclude that we can not use the matrices U and V directly as factor matrices. At the same time, we should not use the matrices Q_U and Q_V directly due to the bad decay of the coefficients in the factor matrices. In the following, we propose two possible approaches to solve this issue, which lead to approximations as accurate as the approximation $Q_U\tilde{C}Q_V^T$. Both approaches can be generalized to higher dimensional settings.

Storing the R factors in addition to the core tensor The first approach is to compute the approximation $T \approx Q_U \tilde{C} Q_V^T$. We now insert the matrices R_U, R_V to obtain an approximation of the form $T \approx U R_U^{-1} \tilde{C} R_V^{-T} V^T$, which contains the nicely decaying factor matrices U, V. This can be seen as approximation with core matrix $\hat{C} = R_U^{-1} \tilde{C} R_V^{-T}$, but we can not compute $\hat{C}$ explicitly due to the issues described in Section 3.4.1. We found that we can instead store R_U, R_V separately. This allows us to accurately evaluate the approximation by using the following order of operations $((FU)R_U^{-1})\tilde{C}(R_V^{-T}(V^T F^T))$. At the same time, storing the additional terms R_U, R_V is the main disadvantage of the approach since we can no longer explicitly form the approximation (2.24).

Rescaling of the Q factors Starting from the approximation of the form $T \approx Q_U \tilde{C} Q_V^T$, we can modify the coefficient decay by introducing diagonal scaling matrices $D_U, D_V \in \mathbb{R}^{r \times r}$. This leads to an approximation of the form $T \approx \bar{Q}_U \bar{C} \bar{Q}_V^T$, where $\bar{Q}_U = Q_U D_U$, $\bar{Q}_V = Q_V D_V$ and $\bar{C} = D_U^{-1} C D_V^{-1}$. We found that setting $(D_U)_{k,k} = \varepsilon / \max(\varepsilon, \min(|F Q_U(:,k)|))$ and$(D_V)_{k,k} = \varepsilon / \max(\varepsilon, \min(|F Q_V(:,k)|))$ for $k = 1, \ldots, r$ leads to a final approximation containing Chebfuns with coefficients $F\bar{Q}_U$ and $F\bar{Q}_V$, which satisfy the Chebfun heuristic [16] and can, thus, be considered proper Chebfun objects; see Figure 3.9(d).

We would like to point out that this rescaling approach does not lead to a loss in accuracy when evaluating the resulting approximation. As in Chebfun2, we obtain an evaluation error of order 10^{-13}. Also note that if we were to set all coefficients in $\bar{Q}_U$ with absolute value smaller than 10^{-13} to zero, we would still achieve an evaluation error of order 10^{-12}.

3.4.3 Incorporation of Chebfun3F in Chebfun

In Chebfun3F we can avoid the issue, discussed in Section 3.4, that the coefficients of the factor matrices do not decay by generalizing the idea of rescaling the Q factors described in Section 3.4.2. For this purpose, we modify Phase 3 in Algorithm 5 in the following way. We set $U = D_U Q_U$, $V = D_V Q_V$, $W = D_W Q_W$ and $\mathcal{C} = \mathcal{T}_f(I, J, K) \times_1 D_U Q_U(I,:)^{-1} \times_2 D_V Q_V(J,:)^{-1} \times_3 D_w Q_W(K,:)^{-1}$, where the diagonal matrices D_U, D_V, D_W are defined as $(D_U)_{i,i} = \varepsilon / \max(\varepsilon, \min(|F^{(1)} Q_U(:,i)|))$, $(D_V)_{j,j} = \varepsilon / \max(\varepsilon, \min(|F^{(2)} Q_V(:,j)|))$ and $(D_W)_{k,k} = \varepsilon / \max(\varepsilon, \min(|F^{(3)} Q_W(:,k)|))$ for $i = 1, \ldots, r_1$, $j = 1, \ldots, r_2$, $k = 1, \ldots, r_3$. Repeating the experiment presented in Figure 3.8 with this modified version of Chebfun3F yields the coefficient decay depicted in Figure 3.10. Observe that the coefficients now decay to almost machine precision as desired.

In collaboration with Nick Trefethen, Nick Hale, Behnam Hashemi and Alex Townsend, we incorporated this modified version of the Chebfun3F algorithm as alternative constructor

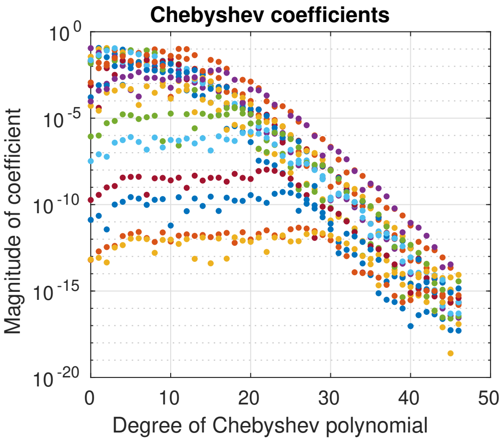


Figure 3.10 – We plot the decay of the univariate Chebyshev interpolation coefficients (2.13) of the functions $u_i^{(1)}$ in (2.24) for the approximation of $f(x, y, z) = \sin(\exp(x + y + z))$ computed using the modified version of Chebfun3F presented in Section 3.4.3.

for Chebfun3 objects into the official Chebfun package. Our alternative constructor can be called using `chebfun3(f,'chebfun3f')` and returns a standard Chebfun3 object.

Remark 3.5. *As in Chebun3 [163], we included a coefficient truncation [16] in the modified version of Chebfun3F. Such a truncation is not included in the original version of Chebfun3F presented in Algorithm 5. This causes the difference in the polynomial degrees in Figure 3.8 and 3.10.*

3.5 Additional insights

3.5.1 Approximations in Lebesgue spaces

In Remark 2.12, we discuss how an approximation of the evaluation tensor (2.18) affects the L^1-norm of the corresponding interpolants. In the following, we study this relationship for the L^2-norm.

Let $p : [-1, 1]^3 \to \mathbb{R}$ be defined based on the evaluation tensor $\mathcal{T} \in \mathbb{R}^{(n+1)\times(n+1)\times(n+1)}$ as in (2.16). Let $\hat{p}$ be defined as in (2.21) based on the tensor $\hat{\mathcal{T}} \in \mathbb{R}^{(n+1)\times(n+1)\times(n+1)}$. By using tensorized Clenshaw-Curtis quadrature as in Remark 2.12 we obtain

$$\begin{aligned}\|p - \hat{p}\|_{L^2}^2 &= \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} (p(x, y, z) - \hat{p}(x, y, z))^2 dxdydz \\ &= \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=0}^{n} \mathcal{W}_{i,j,k}(T_{i,j,k} - \hat{T}_{i,j,k})^2 + \varepsilon_{\mathsf{quad}},\end{aligned}$$

where $\varepsilon_{\mathsf{quad}}$ denotes the quadrature error. The quadrature error occurs, since $(p - \hat{p})^2$ is a polynomial of degree up to $(2n, 2n, 2n)$, but Clenshaw-Curtis quadrature is only exact up to degree (n, n, n) [325]. However, when the entries in the coefficients tensors $\mathcal{A}, \hat{\mathcal{A}}$ defined as in (2.19) based on $\mathcal{T}, \hat{\mathcal{T}}$ decay sufficiently quickly, we can assume that $\varepsilon_{\mathsf{quad}}$ is negligible [27]. Under this strong assumption, we have

$$\|p - \hat{p}\|_{L^2}^2 \approx \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=0}^{n} \mathcal{W}_{i,j,k}((\mathcal{T} - \hat{\mathcal{T}})_{i,j,k})^2 = \|\sqrt{\mathcal{W}} * (\mathcal{T} - \hat{\mathcal{T}})\|_F^2.$$

In order to obtain a good approximation of p in the format (2.24) with $\mathcal{C} \in \mathbb{R}^{r \times r \times r}$, we try to minimize $\|\sqrt{\mathcal{W}} * (\mathcal{T} - \hat{\mathcal{T}})\|_F^2$. Analogous to Remark 2.12, we can achieve this by computing an approximation of $\sqrt{\mathcal{W}}\mathcal{T}$ in Tucker format (2.5). Let $\tilde{\mathcal{T}}_{\mathsf{HOSVD}}$ be obtained by applying a truncated HOSVD [88] to $\sqrt{\mathcal{W}}\mathcal{T}$. Let

$$(\hat{\mathcal{T}}_{\mathsf{HOSVD}})_{i_1,\dots,i_d} = \begin{cases} (\tilde{\mathcal{T}}_{\mathsf{HOSVD}})_{i_1,\dots,i_d} / \sqrt{\mathcal{W}_{i_1,\dots,i_d}} & \mathcal{W}_{i_1,\dots,i_d} \neq 0 \\ (\tilde{\mathcal{T}}_{\mathsf{HOSVD}})_{i_1,\dots,i_d} & \mathcal{W}_{i_1,\dots,i_d} = 0. \end{cases}$$

for $i_\ell = 0, \dots, n,\ \ell = 1, \dots, d$. Using the tensor $\hat{\mathcal{T}}_{\mathsf{HOSVD}}$ to define $\hat{p}$ yields a close to optimal approximation of p in the L^2-norm. Note that $\hat{\mathcal{T}}_{\mathsf{HOSVD}}$ has multilinear rank at most (r, r, r) and can thus be represented in Tucker format with with $\mathcal{C} \in \mathbb{R}^{r \times r \times r}$.

In Figure 3.11 we demonstrate that the singular values of $(\sqrt{\mathcal{W}}\mathcal{T})^{\{1\}}$ are closely related to the error $\|p - \hat{p}\|_{L^2}$. Surprisingly, both applying the HOSVD directly to $\mathcal{T}$ and using the tensor $\hat{\mathcal{T}}_{\mathsf{HOSVD}}$ yield polynomials with almost identical errors. However, it is not possible to precisely determine the accuracy of $\|p - \hat{p}\|_{L^2}$ based on the singular values of $\hat{\mathcal{T}}^{\{1\}}$. We conclude that the we can accurately estimate the multilinear rank required for accurate approximations of the form (2.24) with respect to the L^2-norm by truncating the singular values of ℓ-mode matricizations of $\sqrt{\mathcal{W}}\mathcal{T}$ for sufficiently large n. For computing an approximation with fixed multilinear rank it does not seem to be necessary to take the weight tensor $\sqrt{\mathcal{W}}$ into account.

3.5.2 Further compression

In this section, we want to gain additional insight into the impact of the size of the core tensor on approximations of the form (2.24). Moreover, we study if it is possible to further compress the Tucker approximation (2.5) of the evaluation tensor (2.18). This section is inspired by Rachel Minster's talk [229] on the paper [278] at SIAM-ALA 2021.

We are interested in evaluating a kernel function $k : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ at all pairs $(x^{(i)}, y^{(j)})$ for $i = 1, \dots, M,\ j = 1, \dots, N$, where $x^{(i)} \in [a, b]^2,\ y^{(j)} \in [c, d]^2,\ a, b, c, d \in \mathbb{R}$ such that $[a, b] \cap [c, d] = \emptyset$. Let $K \in \mathbb{R}^{M \times N}$ denote the matrix of evaluations, i.e. $K_{i,j} = k(x^{(i)}, y^{(j)})$. The direct computation of K requires $M \cdot N$ kernel evaluations. In the following, we propose a method to approximate K using fewer kernel evaluations.

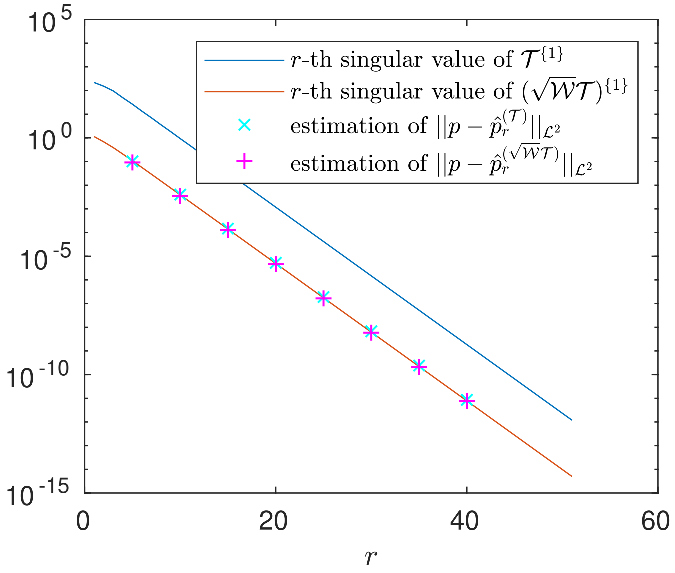


Figure 3.11 – Let $\mathcal{T}$ denote the sample tensor of $f(x,y,z) = \cosh 3(x+y+z)^{-1}$ on a $75 \times 75 \times 75$ Chebyshev grid. We depict the singular value decay of $\mathcal{T}^{\{1\}}$ and of $(\sqrt{\mathcal{W}}\mathcal{T})^{\{1\}}$. The interpolant (2.16) based on $\mathcal{T}$ is denoted by p. The interpolant based $\hat{\mathcal{T}}_{\mathsf{HOSVD}}$ with core tensor in $\mathbb{R}^{r\times r\times r}$ is denoted by $\hat{p}_r^{(\sqrt{\mathcal{W}}\mathcal{T})}$ and the interpolant obtained from a direct HOSVD of $\hat{\mathcal{T}}$ with core tensor in $\mathbb{R}^{r\times r\times r}$ is denoted by $\hat{p}_r^{(\mathcal{T})}$. We plot estimations of $||p-\hat{p}_r^{(\sqrt{\mathcal{W}}\mathcal{T})}||_{L^2}$ and $||p-\hat{p}_r^{(\mathcal{T})}||_{L^2}$ computed using 10 000 sample points for different values of r. Note that the estimated L^2-errors almost coincide for the same values of r.

We proceed by approximating k as a function using tensorized interpolation (2.16) and approximations of the coefficient tensor (2.22) to obtain an approximation of the form

$$k(x_1,x_2,y_1,y_2) \approx p(x_1,x_2,y_1,y_2) = \sum_{i=0}^{n_1}\sum_{j=0}^{n_2}\sum_{k=0}^{n_3}\sum_{l=0}^{n_4} \hat{\mathcal{A}}_{i,j,k,l} T_i(x_1)T_j(x_2)T_k(y_1)T_l(y_2).$$

where $\hat{\mathcal{A}}$ is derived from an approximation $\hat{\mathcal{T}} \in \mathbb{R}^{(n_1+1)\times\cdots\times(n_4+1)}$ of the evaluation tensor (2.18). Given $\hat{\mathcal{T}}$ we can compute the approximation

$$\tilde{K} = E_x(F^{(2)} \otimes F^{(1)})\hat{\mathcal{T}}^{<2>}(F^{(4)} \otimes F^{(3)})E_y^T, \tag{3.15}$$

of the kernel matrix K, where $\otimes$ denotes the Kronecker product, $F^{(\ell)}$ is defined as in Equation (2.14), and $E_x \in \mathbb{R}^{M\times(n_1+1)(n_2+1)}$ and $E_y \in \mathbb{R}^{N\times(n_3+1)(n_4+1)}$ encode the evaluations of the Chebyshev polynomials at the points of interest, i.e.

$$(E_x)_{i,j+k(n_1+1)} = T_j(x_1^{(i)}) \cdot T_k(x_2^{(i)}) \text{ for } i=1,\ldots,M,\ j=0,\ldots,n_1,\ k=0,\ldots,n_2,$$
$$(E_y)_{i,j+k(n_1+1)} = T_j(y_1^{(i)}) \cdot T_k(y_2^{(i)}) \text{ for } i=1,\ldots,N,\ j=0,\ldots,n_3,\ k=0,\ldots,n_4.$$

Note that multiplying the kernel approximation (3.15) with a vector requires $\mathcal{O}((M+$

$N)n^2 + n^4)$ operations when $n_1 = n_2 = n_3 = n_4 = n$. This is cheaper than multiplying K directly with a vector when $(M+N) \gg n^2$.

Remark 3.6. *Note that a generalization of these ideas to kernels* $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *for arbitrary* $d \in \mathbb{N}$ *is straightforward, but generalizing Equation* (3.15) *is only beneficial for matrix vector products when* $(M+N) \gg n^d$.

In [229], it is suggested to obtain $\hat{\mathcal{T}}$ in Tucker format (2.5) using a non-adaptive approach similar to Algorithm 5. First, the span of each matrizisation of $\mathcal{T}$ is approximated to obtain the factor matrices. For this purpose the randomized row interpolatory decomposition algorithm (RRID) [156] (see Algorithm 6) is applied to matricizations of subtensors of $\mathcal{T}$. The RRID algorithm immediately yields index sets that can be used to define the core tensor of the Tucker approximation using oblique projections. The overall kernel approximation algorithm is formalized in Algorithm 7.

Algorithm 6 Randomized row interpolatory decomposition

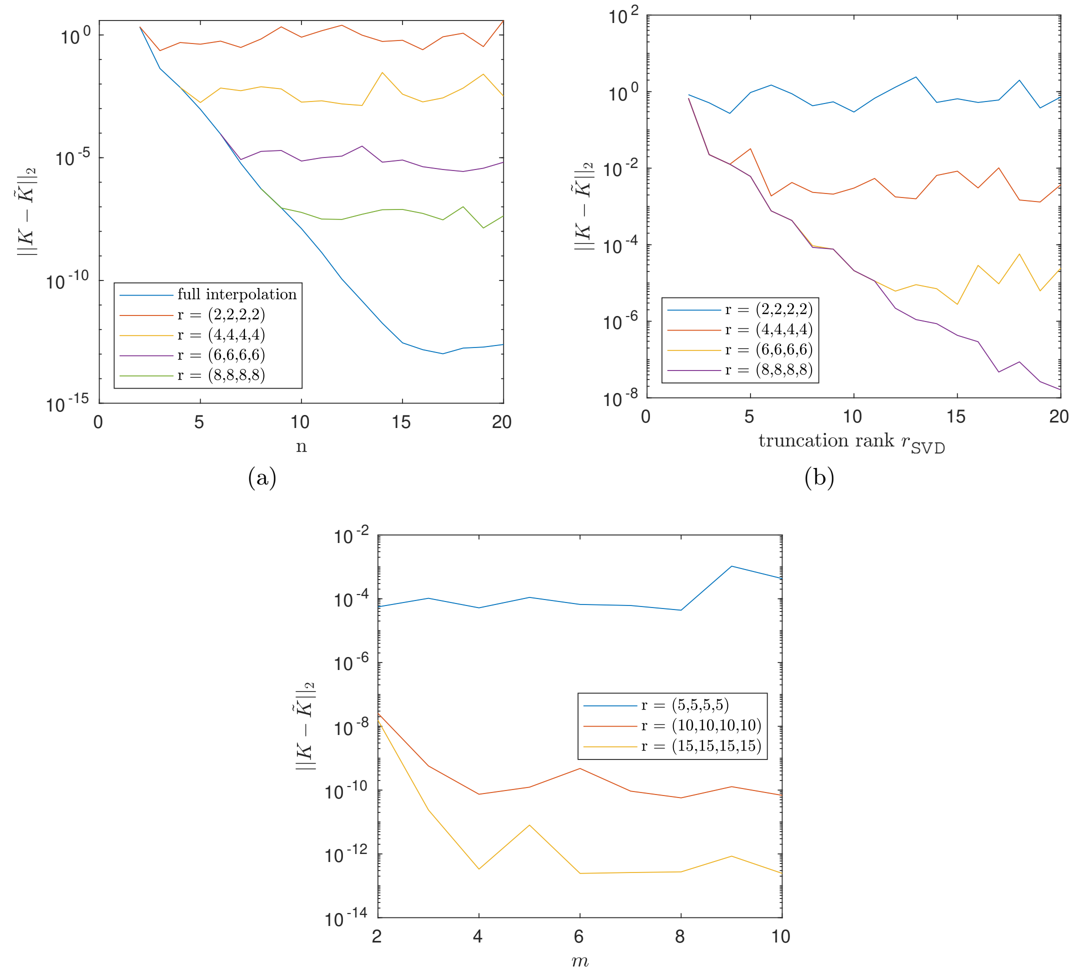
```
Input: matrix $M \in \mathbb{R}^{m \times n}$, target rank $r$, oversampling parameter $p$
Output: approximation $M \approx UM(J,:)$
sample a Gaussian random matrix $\Omega \in \mathbb{R}^{n \times (r+p)}$
compute a thin QR factorization $QR = M\Omega$
$Q = Q(:, 1:r)$
apply a pivoted thin QR factorization to $Q^T$ and let $J$ denote the set of pivot elements
$U = Q(Q(J,:))^{-1}$
```

Algorithm 7 Kernel approximation

```
Input: kernel function $k : [a,b]^2 \times [c,d]^2 \to \mathbb{R}$, polynomial degree $(n_1, n_2, n_3, n_4)$,
    size of the core tensor $r_1 \times r_2 \times r_3 \times r_4$, index sets $I_\ell \subseteq \{0, \dots, n_\ell\}$ for $1 \le \ell \le 4$
Output: approximation $\hat{\mathcal{T}}$ of the evaluation tensor (2.18) in Tucker format (2.5)
Let $\mathcal{T}$ denote the tensor containing the evaluations of $k$ on a $(n_1+1) \times (n_2+1) \times$
    $(n_3+1) \times (n_4+1)$ Chebyshev grid mapped onto the domain $[a,b]^2 \times [c,d]^2$.
Let $M_1 = T(:, I_2, I_3, I_4)^{\{1\}}$, $M_2 = T(I_1, :, I_3, I_4)^{\{2\}}$, $M_3 = T(I_1, I_2, :, I_4)^{\{3\}}$, $M_4 =$
    $T(I_1, I_2, I_3, :)^{\{4\}}$.
for $\ell = 1, \dots, 4$
    Apply RRID with target rank $r_\ell$ to obtain the approximation $M_\ell \approx U_\ell M_\ell(J_\ell, :)$ .
$\hat{\mathcal{T}} = \mathcal{T}(J_1, J_2, J_3, J_4) \times_1 U_1 \times_2 U_2 \times_3 U_3 \times_4 U_4$
```

In the following numerical experiment, we consider the kernel function $k(x,y) = \frac{1}{\|x-y\|_2}$. We sample x_i uniformly from $[0,1]^2$ for $1 \le i \le M = 200$ and y_j uniformly from $[2,3]^2$ for $1 \le j \le N = 150$. In Figure 3.12(a) we study the approximation error $\|K - \tilde{K}\|_2$ for approximations obtained using Equation (3.15), where $\hat{\mathcal{T}}$ is computed using Algorithm 7 with different prescribed core tensor sizes. We observe that error of tensorized interpolation (2.16) with polynomial degree (r,r,r,r) without any approximation of the evaluation tensor yields the same approximation error as Algorithm 7 with core tensor in $\mathbb{R}^{r \times r \times r \times r}$ for any polynomial degree (n,n,n,n) with $n > r$. This implies that the error is dominated by the chosen core tensor size for this kernel function. Figure 3.12(b) shows that the matrix $\tilde{\mathcal{T}}^{<2>}$ used in (3.15) does not seem to be full rank. Truncating the SVD of $\tilde{\mathcal{T}}^{<2>}$ with rank larger than $2r$ does not improve the approximation error when the core tensor is in $\mathbb{R}^{r\times r\times r\times r}$. Thus, it is possible to further compress the approximation (3.15). So far, we did not analyze the impact of the index sets $I_1, \ldots, I_4$ in Algorithm 7. In Figure 3.12(c), we observe that choosing $m > 2$ indices in each of the index sets $I_1, \ldots, I_4$ does not improve the error for core tensors in $\mathbb{R}^{5\times 5\times 5\times 5}$, i.e. the $m^3 = 8$ randomly selected fibers in each mode suffice to find suitable factor matrices. For core tensors in $\mathbb{R}^{10\times 10\times 10\times 10}$ and $\mathbb{R}^{15\times 15\times 15\times 15}$, the approximation error decreases for m increasing from 2 to 4 and stagnates afterwards. This demonstrates, that we only need slightly more than r_ℓ fibers to identify good factor matrices for a given r_ℓ. At the same time, it confirms that we do not need to fully evaluate $\mathcal{T}$ to compute suitable factor matrices.

We conclude this section by summarizing our observations for general functional Tucker approximations of the form (2.24). Note that these findings also apply to Chebfun3 and Chebfun3F. While it can be beneficial to use a smaller core tensor compared to the multilinear rank as demonstrated in Section 5.5.1, we showed that this is not necessarily the case for all functions as demonstrated in Figure 3.12(a). A functional Tucker approximation of the form (2.24) can potentially be compressed further to minimize the required storage. This motivates the algorithms developed in Chapter 4, where we compress approximations (2.24) further by computing TT approximations of the core tensor. Lastly, we observe in Figure 3.12(c) that randomly selected subtensors (as used in Algorithm 4) tend to be sufficient to compute factor matrices that lead to accurate Tucker decompositions.



(a)

(b)

(c)

Figure 3.12 – Comparison of $\|K-\tilde{K}\|_2$ for the setting in Section 3.5.2. Top Left: The matrix $\tilde{K}$ (3.15) is computed based on $\tilde{\mathcal{T}}$ obtained using Algorithm 7 with polynomial degree (n,n,n,n) and core tensor in $\mathbb{R}^{r_1\times r_2\times r_3\times r_4}$ as shown in the legend. The index sets in Algorithm 7 are set to $I_\ell=\{0,\dots,n_\ell\}$ for $\ell=1,\dots,4$. Additionally plot the error when $\tilde{\mathcal{T}}$ is computed using tensorized interpolation (2.16) without any approximation of the evaluation tensor (full interpolation). Top Right: The tensor $\tilde{\mathcal{T}}$ obtained using Algorithm 7 with polynomial degree $(20,20,20,20)$ and core tensor in $\mathbb{R}^{r_1\times r_2\times r_3\times r_4}$ as shown in the legend. The index sets in Algorithm 7 are set to $I_\ell=\{0,\dots,20\}$ for $\ell=1,\dots,4$. We then compute a truncated SVD of $\tilde{\mathcal{T}}^{<2>}$ with truncation rank r_{SVD}. We compute the matrix $\tilde{K}$ (3.15) based on this compressed version of $\tilde{\mathcal{T}}^{<2>}$. Bottom: The matrix $\tilde{K}$ is computed using Algorithm 7 with polynomial degree $(20,20,20,20)$ and core tensor in $\mathbb{R}^{r_1\times r_2\times r_3\times r_4}$ as shown in the legend. The index sets I_ℓ each contain m indices uniformly sampled from $\{0,\dots,20\}$.

4 Extended functional tensor train approximation

This chapter is concerned with the notoriously difficult task of approximating multivariate functions on the tensor product domain $[-1,1]^d$ for large values of d. In principle, we could extend the algorithms presented in Chapter 3 to functions depending on more than three variables. However, the functional Tucker format (2.11) is not well suited for the high-order tensors arising from the evaluation of a function in many variables due to the size of the core tensor scaling exponentially with respect to d. Other formats, such as the functional tensor train (FTT) format (2.12) and the extended functional tensor train (EFTT) format proposed in this chapter, are better suited for this purpose and will require different construction algorithms.

As described in Section 2.3.4, approximations in the FTT format can be obtained by combining tensorized Chebyshev interpolation (2.16) with Algorithm 3 to compute a tensor train decomposition (2.6) of the coefficient tensor (2.19) [41]. A slightly different approach to obtain FTT approximations is proposed in [141], where a continuous version of the TT-cross algorithm [251] is applied directly to the function. All univariate functions occurring in this process are discretized using parameterizations such as basis expansions. In principle, every univariate function could be parameterized differently. In the special case that univariate interpolation with the same basis functions is used to discretize all univariate functions in the same mode, this approach is equivalent to a TT approximation of the coefficient tensor for tensorized interpolation.

The FTT format (2.12) requires the storage of $R_{\ell-1} \cdot R_\ell$ univariate functions for each $\ell = 1, \ldots, d$. These functions (or their parametrizations) will often be linearly dependent. In this case, we can compress the format further by storing these functions in terms of linear combinations of $r_\ell \ll R_{\ell-1} \cdot R_\ell$ basis functions. The resulting approximation can be seen as a variant of the hierarchical Tucker format proposed in [152]. When the approximation is constructed from a low-rank approximation of the coefficient tensor, we can obtain such a compressed format by following the ideas of Khoromskij [186]. There a two-level Tucker format is proposed, which is obtained by first computing a

Tucker approximation [327] before approximating the core tensor in $\mathbb{R}^{r_1\times\cdots\times r_d}$ further. Computing a TT approximation of the core tensor leads to a so-called extended TT format [99, 108, 286], from which we can derive a functional low-rank approximation in the EFTT format. This EFTT approximation corresponds to a compressed FTT approximation.

In this chapter, we propose a novel algorithm to efficiently compute functional low-rank approximations using the EFTT format. Our algorithm is based on first obtaining suitable factor matrices for a Tucker approximation (2.5) of the coefficient tensor. The columns of these matrices are determined by applying a combination of adaptive cross approximation [30] with randomized pivoting to matricizations of the coefficient tensor. We then apply oblique projections based on discrete empirical interpolation as in Section 3.2 to implicitly construct a suitable core tensor for the Tucker approximation. In a second step, we compute a TT approximation of this core tensor using Algorithm 3. The combination of the Tucker and TT approximation yields the desired functional low-rank approximation. The main advantage of our approach compared to a direct TT approximation of the coefficient tensor is that the approximated core tensor is potentially much smaller than the coefficient tensor. This reduces the storage complexity of the approximation and it significantly reduces the number of function evaluations needed in the computation of the TT approximation, which we demonstrate in our numerical experiments.

This chapter is based on the article [306]. Its remainder is structured as follows. In Section 4.1, we define the EFTT format. Our novel algorithm to compute approximations in the EFTT format is presented in Section 4.2. Our numerical experiments in Section 4.3 demonstrate the advantages of our novel algorithm compared to a direct TT approximation of the coefficient tensor as in [41] and to the FTT algorithm in [141]. The test functions for our numerical experiments are defined in the Appendix 4.A.

4.1 Extended functional tensor train format

A function $f : [-1,1]^d \to \mathbb{R}$ is said to be represented in EFTT format with TT representation ranks $(R_1,\dots,R_{d-1}) \in \mathbb{N}^{d-1}$ and multi-linear representation ranks $(r_1,\dots,r_d) \in \mathbb{N}^d$ when it is given in terms of univariate functions $u_j^{(\ell)} : [-1,1] \to \mathbb{R}$ for $j = 1,\dots,r_\ell$, $\ell = 1,\dots,d$ and TT cores $\mathcal{H}^{(\ell)} \in \mathbb{R}^{R_{\ell-1}\times r_\ell \times \mathbb{R}_\ell}$ for $\ell = 1,\dots,d$ as

$$f(x_1,\dots,x_d) \approx \sum_{j_1=1}^{r_1}\cdots\sum_{j_d=1}^{r_d}\sum_{\alpha_1=1}^{R_1}\cdots\sum_{\alpha_{d-1}}^{R_{d-1}} \mathcal{H}^{(1)}_{1,j_1,\alpha_1}\cdots\mathcal{H}^{(d)}_{\alpha_{d-1},j_d,1} u^{(1)}_{j_1}(x_1)\cdots u^{(d)}_{j_d}(x_d), \quad (4.1)$$

where we formally set $R_0 = R_d = 1$ to simplify notation.

Following the construction in Section 2.3.4 we can derive approximations in the EFTT format (4.1) from low-rank approximations of the evaluation tensor $\mathcal{T}$ (2.18). We first compute a Tucker decomposition (2.5) of $\mathcal{T} \in \mathbb{R}^{(n_1+1)\times\cdots\times(n_d+1)}$ of the form

$$\mathcal{T} \approx \mathcal{C} \times_1 U^{(1)} \times_2 \cdots \times_d U^{(d)}, \tag{4.2}$$

with $\mathcal{C} \in \mathbb{R}^{r_1\times\cdots\times r_d}$ and $U^{(\ell)} \in \mathbb{R}^{(n_\ell+1)\times r_\ell}$. In a second step, the core tensor $\mathcal{C}$ is approximated in TT format (2.6) by

$$\hat{\mathcal{C}}_{i_1,\ldots,i_d} = \sum_{\alpha_1=1}^{R_1} \cdots \sum_{\alpha_{d-1}=1}^{R_{d-1}} \mathcal{H}^{(1)}_{1,i_1,\alpha_1} \mathcal{H}^{(2)}_{\alpha_1,i_1,\alpha_2} \cdots \mathcal{H}^{(d-1)}_{\alpha_{d-2},i_{d-1},\alpha_{d-1}} \mathcal{H}^{(d)}_{\alpha_{d-1},i_d,1} \tag{4.3}$$

for $i_\ell = 1,\ldots,r_\ell,\ \ell = 1,\ldots,d$, with the TT cores $\mathcal{H}^{(\ell)} \in \mathbb{R}^{R_{\ell-1}\times r_\ell \times R_\ell}$. Inserted into (4.2), this yields an approximation of the evaluation tensor in the extended TT format [286] of the form

$$\hat{\mathcal{T}}_{i_1,\ldots,i_d} = \sum_{j_1=1}^{r_1} \cdots \sum_{j_d=1}^{r_d} \sum_{\alpha_1=1}^{R_1} \cdots \sum_{\alpha_{d-1}}^{R_{d-1}} \mathcal{H}^{(1)}_{1,j_1,\alpha_1} \cdots \mathcal{H}^{(d)}_{\alpha_{d-1},j_d,1} U^{(1)}_{i_1,j_1} \cdots U^{(d)}_{i_d,j_d}, \tag{4.4}$$

for $i_\ell = 0,\ldots,n_\ell,\ \ell = 1,\ldots,d$. This only requires $\mathcal{O}(drR^2 + dnr)$ storage, where $R = \max R_\ell$, $r = \max r_\ell$, $n = \max n_\ell$, which compares favorably with the $\mathcal{O}(dnR^2)$ storage needed by a direct TT approximation (2.6) of $\mathcal{T}$, especially when $r \ll n$. In Figure 4.1, we visualize the corresponding approximation of the coefficient tensor (2.19) given by

$$\hat{\mathcal{A}}_{i_1,\ldots,i_d} = \sum_{k_1=0}^{n_1} \cdots \sum_{k_d=0}^{n_d} \sum_{j_1=1}^{r_1} \cdots \sum_{j_d=1}^{r_d} \sum_{\alpha_1=1}^{R_1} \cdots \sum_{\alpha_{d-1}}^{R_{d-1}} \mathcal{X}_{k_1,\ldots,k_d,j_1,\ldots,j_d,\alpha_1,\ldots,\alpha_{d-1}}, \tag{4.5}$$
$$\mathcal{X}_{k_1,\ldots,k_d,j_1,\ldots,j_d,\alpha_1,\ldots,\alpha_{d-1}} = \mathcal{H}^{(1)}_{1,j_1,\alpha_1} \cdots \mathcal{H}^{(d)}_{\alpha_{d-1},j_d,1} U^{(1)}_{i_1,j_1} \cdots U^{(d)}_{i_d,j_d} F^{(1)}_{k_1,i_1} \cdots F^{(d)}_{k_d,i_d},$$

for $i_\ell = 0,\ldots,n_\ell,\ \ell = 1,\ldots,d$. Inserting $\hat{\mathcal{A}}$ into (2.21) yields the desired approximation (4.1) with TT cores $H^{(\ell)}$ and with

$$u^{(\ell)}_j(x) := \sum_{i=0}^{n_\ell} \sum_{k=0}^{n_\ell} F^{(\ell)}_{k,i} U^{(\ell)}_{i,j} T_k(x).$$

The resulting approximation error is bounded by Lemma 2.2. When we store the tensors $\mathcal{H}^{(\ell)}$ and the matrices $(F^{(\ell)}U^{(\ell)}) \in \mathbb{R}^{(n_\ell+1)\times r_\ell}$ separately, we can compute the point evaluation of (4.1) at $(x_1,\ldots,x_d) \in [-1,1]^d$ by first contracting $F^{(\ell)}U^{(\ell)}$ with the vectors $(T_0(x_\ell),\ldots,T_{n_\ell}(x_\ell))$ for $\ell = 1,\ldots,d$ and by then contracting the resulting vectors with the TT cores $\mathcal{H}^{(\ell)}$ [247]. This requires $\mathcal{O}(drn + drR^2)$ operations, where $R = \max R_\ell$, $r = \max r_\ell$, $n = \max n_\ell$.

Remark 4.1. *Note that EFTT approximations obtained from evaluation tensor approximations of the form* (4.4) *are closely related to FTT approximations. Given an approximation* (4.4)*, we can define the TT cores* $\mathcal{G}^{(\ell)} = \mathcal{H}^{(\ell)} \times_2 U^{(\ell)}$*. The TT approxima-*

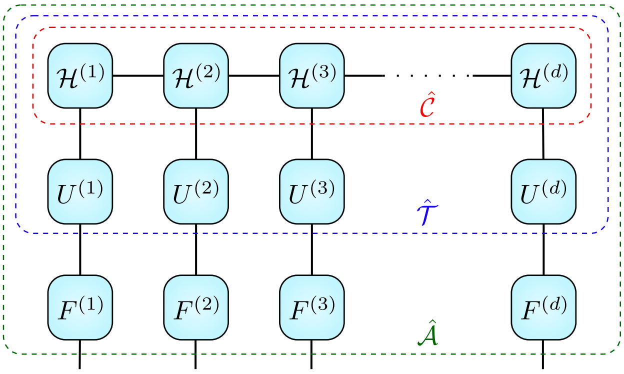


Figure 4.1 – Tensor network representation [247] of the coefficient tensor $\hat{\mathcal{A}}$ in (4.5) corresponding to EFTT format (4.1). The colored boxes mark the subtensors corresponding to the approximation of the evaluation tensor $\hat{\mathcal{T}}$ and the approximation of the core tensor $\hat{\mathcal{C}}$.

tion (2.6) *given by the cores* $\mathcal{G}^{(\ell)}$ *can be used to derive an equivalent FTT approximation using the construction in Section 2.3.4. Conversely, EFTT approximations can be seen as compression of such a FTT approximation, since the TT cores are stored in terms of* $\mathcal{H}^{(\ell)}$ *and* $U^{(\ell)}$ *requiring only* $\mathcal{O}((n_\ell+1)r_\ell + R_{\ell-1}r_\ell R_\ell)$ *storage instead of* $\mathcal{O}(R_{\ell-1}(n_\ell+1)R_\ell)$ *for the uncompressed core* $\mathcal{G}^{(\ell)}$*.*

4.2 Approximation algorithm

In the following, we develop a novel algorithm for computing approximations in the EFTT format (4.1). Our algorithm obtains the factor matrices $U^{(1)}, \dots, U^{(d)}$ for the Tucker approximation (4.2) from fibers of the evaluation tensor $\mathcal{T}$ via a variant of column subset selection [79, 93]. Following [100], the core tensor $\mathcal{C}$ is obtained as a subtensor of $\mathcal{T}$ by applying discrete empirical interpolation. Thus, there is no need to form $\mathcal{C}$ explicitly and we apply a variant of the TT-cross algorithm [251] to compute the TT cores $\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(d)}$ for the TT approximation (4.3) from only some entries of $\mathcal{C}$.

Fiber identification. Applying cross approximation (2.1) to $\mathcal{T}^{\{\ell\}}$ yields index sets $\hat{I}_\ell, \hat{J}_\ell$ that determine an approximation of the form

$$\mathcal{T}^{\{\ell\}} \approx \mathcal{T}^{\{\ell\}}(:, \hat{J}_\ell)(\mathcal{T}^{\{\ell\}}(\hat{I}_\ell, \hat{J}_\ell))^{-1}\mathcal{T}^{\{\ell\}}(\hat{I}_\ell, :). \tag{4.6}$$

Note that the matrix $\mathcal{T}^{\{\ell\}}(:, \hat{J}_\ell)$ contains mode-ℓ fibers of $\mathcal{T}$. If the approximation error of (4.6) is small, we choose $\hat{U}^{(\ell)} = \mathcal{T}^{\{\ell\}}(:, \hat{J}_\ell)$ as an approximate basis of mode-ℓ fibers. Due to the large size of $\mathcal{T}^{(\ell)}$, it might not be feasible to apply either ACA with full or ACA with partial pivoting as described in Algorithm 1 and Remark 2.1. Inspired by the success of randomization in numerical linear algebra [222, 339], we propose to determine the next indices in line 5 of Algorithm 1 based on sampling. This leads to Algorithm 8, which samples a fixed number s of random entries and picks the entry of largest absolute value. These random entries are also used for stopping the algorithm and to determine r_ℓ adaptively. Note that the update in line 10 is only performed implicitly and the subtraction is carried out each time an entry of B is evaluated. Overall, Algorithm 8 applied to $\mathcal{T}^{\{\ell\}}$ determines the index sets $\hat{I}_\ell, \hat{J}_\ell$ using only $\mathcal{O}(r_\ell^3 + s r_\ell^2)$ entries of $\mathcal{T}$. Afterwards, we explicitly compute $\hat{U}^{(\ell)} = \mathcal{T}^{\{\ell\}}(:, \hat{J}_\ell)$ by evaluating $n_\ell r_\ell$ entries of $\mathcal{T}$. We perform the described procedure for every $\ell = 1, \dots, d$ to determine $\hat{U}^{(1)}, \dots, \hat{U}^{(d)}$; see also lines 4 and 5 of Algorithm 9 below.

Algorithm 8 Adaptive cross approximation with randomized pivoting

```
1:  Input: procedure to evaluate entries of $A \in \mathbb{R}^{m \times n}$, tolerance $\varepsilon$, number of samples $s$
2:  Output: index sets $I$ and $J$ such that $A \approx A(:,J)A(I,J)^{-1}A(I,:)$
3:  $I = \emptyset$, $J = \emptyset$
4:  $B = A$
5:  while true
6:      Construct $S \subset \{1,\dots,n\} \times \{1,\dots,m\}$ by uniformly sampling $s$ index pairs $(i,j)$.
7:      if $\max_{(i,j)\in S} |B_{i,j}| \le \varepsilon$ return $I, J$ end
8:      $(i^*, j^*) = \arg\max_{(i,j)\in S} |B_{i,j}|$
9:      $I = I \cup \{i^*\}$, $J = J \cup \{j^*\}$
10:     $B = A - A(:,J)A(I,J)^{-1}A(I,:)$
```

Tucker approximation To arrive at the Tucker approximation (4.2) we need to project the fibers of $\mathcal{T}$ onto the spans of $\hat{U}^{(1)}, \dots, \hat{U}^{(d)}$. Analogous to Section 3.2.3, we first compute economic QR decompositions $\hat{U}^{(\ell)} = Q^{(\ell)} R^{(\ell)}$ and apply DEIM (see Algorithm 2) to identify index sets I_ℓ. We then apply oblique projections (2.3) to each mode of $\mathcal{T}$ to compute the Tucker approximation

$$\begin{aligned}\mathcal{T} &\approx (\mathcal{T} \times_1 \Phi_{I_1}^T \times_2 \cdots \times_d \Phi_{I_d}^T) \times_1 Q^{(1)}(\Phi_{I_1}^T Q^{(1)})^{-1} \times_2 \cdots \times_d Q^{(d)}(\Phi_{I_d}^T Q^{(d)})^{-1} \\ &= \mathcal{C} \times_1 U_1 \times_2 \cdots \times_d U_d,\end{aligned}$$

with factor matrices $U^{(\ell)} = Q^{(\ell)}(\Phi_{I_\ell}^T Q^{(\ell)})^{-1}$ and core tensor $\mathcal{C} = \mathcal{T}(I_1, \dots, I_d)$. Note that the subtensor $\mathcal{C}$ of $\mathcal{T}$ will not be explicitly formed; we only need to evaluate some its entries when computing the TT approximation of $\mathcal{C}$. The whole process of computing $U^{(\ell)}$ and $\mathcal{C}$ from $\hat{U}^{(\ell)}$ is summarized in Algorithm 9. Alternative randomized algorithms to compute Tucker approximations have been proposed in [228, 230].

Remark 4.2. *For simplicity, we have assumed that the values of $n_1, \dots, n_d$, determining the size of the expansion* (2.16) *and of $\mathcal{T}$ are given as input. In practice, these values are usually not provided. To ensure that the approximation error of the final approximation* (4.1) *is small, we use the heuristic introduced in Remark 2.11 to determine $n_1, \dots, n_d$ adaptively. We initially set $n_1 = \cdots = n_d = 16$. If applying the chopping heuristic [16] to*

Algorithm 9 Tucker approximation

```
Input: procedure to evaluate entries of $\mathcal{T} \in \mathbb{R}^{(n_1+1)\times\cdots\times(n_d+1)}$, tolerance $\varepsilon$, number of samples $s$
Output: Matrices $U^{\ell} \in \mathbb{R}^{(n_\ell+1)\times r_\ell}$ and a procedure to evaluate entries of $\mathcal{C} \in \mathbb{R}^{r_1\times\cdots\times r_d}$, defining a Tucker approximation (4.2)
for $\ell = 1, \ldots, d$
    $\hat{J}_\ell \leftarrow$ index set $J$ returned by Algorithm 8 applied to $\mathcal{T}^{\{\ell\}}$ with tolerance $\varepsilon$ and $s$ samples.
    $\hat{U}^{(\ell)} = \mathcal{T}^{\{\ell\}}(:, \hat{J}_\ell)$
    Compute economic QR decomposition $\hat{U}^{(\ell)} = Q^{(\ell)} R^{(\ell)}$.
    $I_\ell = \mathrm{DEIM}(Q^{(\ell)})$ (see Algorithm 2)
    $U^{(\ell)} = Q^{(\ell)}(Q^{(\ell)}(I_\ell, :))^{-1}$
$\mathcal{C} = \mathcal{T}(I_1, \ldots, I_d)$
```

$\hat{U}^{(\ell)}$ indicates that n_ℓ is not sufficiently large then one sets $n_\ell = 2n_\ell + 1$, updates $\mathcal{T}$ and repeats the procedure from line 4 to compute a new index set $\hat{J}_\ell$.

TT cores It remains to compute a TT approximation of the core tensor (4.3) to obtain the desired extended TT approximation (4.4). For this purpose, we use the TT toolbox[*] implementation of Algorithm 3. This algorithm is rank-adaptive and requires the evaluation of only $\mathcal{O}(drR^2)$ entries of $\mathcal{C}$ where $R = \max R_\ell$, $r = \max r_\ell$.

EFTT approximation algorithm We formalize the overall procedure for computing approximations in the EFTT format (4.1) in Algorithm 10. Note that $r_1, \ldots, r_d$ and $R_1, \ldots, R_{d-1}$ are determined adaptively in lines 5 and 6. Let $R = \max R_\ell$, $r = \max r_\ell$, $n = \max n_\ell$. The total number of evaluations of f in Algorithm 10 is $\mathcal{O}(dr^3 + dnr + dsr^2 + drR^2)$. In contrast, applying Algorithm 3 directly to $\mathcal{T}$ requires $\mathcal{O}(dnR^2)$ evaluations and yields an approximation in FTT format (2.6).

4.3 Numerical experiments

In this section, we present numerical experiments[†] to study the performance of Algorithm 10. Unless mentioned otherwise the sample size in Algorithm 1 is set to $s = \min\{\bar{n}/2, 50\}$, where $\bar{n} = ((n_1+1)(n_2+1)\cdots(n_d+1))^{1/d}$, and all tolerances are set to 10^{-10}. The approximations error is measured via a Monte Carlo estimation of the relative L^2-error based on function evaluations at 10 000 sample points.

---

[*] The TT toolbox is available from https://github.com/oseledets/TT-Toolbox.

[†] The MATLAB and Python code to reproduce these results is available from https://github.com/cstroessner/EFTT.

Algorithm 10 EFTT approximation

```
1: Input: function $f : [-1,1]^d \to \mathbb{R}$, tolerance $\varepsilon$, number of samples $s$
2: Output: TT cores $\mathcal{H}^{(\ell)}$ and procedures for evaluating univariate functions $u_j^{(\ell)}$
   defining an approximation of $f$ in the EFTT format (4.1)
3: $n_1 = \cdots = n_d = 16$
4: Define a procedure to evaluate entries of the evaluation tensor $\mathcal{T}$ defined in (2.18).
5: Apply Algorithm 9 with tolerance $\varepsilon$ and $s$ samples to $\mathcal{T}$ to determine $\hat{U}^{(\ell)}$. Simultaneously,
   we update $n_1, \ldots, n_d$ as well as $\mathcal{T}$ as described in Remark 4.2.
6: Compute the tensors $\mathcal{H}^{(\ell)}$ by applying Algorithm 3 with tolerance $\varepsilon$ to $\mathcal{C}$.
7: Create procedures to evaluate univariate functions $u_j^{(\ell)}(x) = \sum_{i=0}^{n_\ell} \sum_{k=0}^{n_\ell} F_{k,i}^{(\ell)} U_{i,j}^{(\ell)} T_k(x)$.
```

4.3.1 Comparison to a direct TT approximation

We first compare Algorithm 10 yielding an approximation in the EFTT format (4.1) to a direct TT approximation of the evaluation tensor $\mathcal{T} \in \mathbb{R}^{100\times\cdots\times 100}$ (2.18) using Algorithm 3 as in [41]. The latter approach yields an approximation in the FTT format (2.12). Since a direct TT approximation is not adaptive with respect to the polynomial degree, we fix the polynomial degree throughout this comparison, i.e., we replace line 3 by $n_1 = \cdots = n_d = 100$ and do not update adaptively in line 5 of Algorithm 10.

Benchmark functions. For the set of benchmark functions defined in Appendix 4.A.1, we on average reduce the number of function evaluations required by 30.6% and the required storage by 41.6% when using Algorithm 10 compared to the direct TT approximation of the coefficient tensor. For the Ackley function the reduction is 88.8% in terms of function evaluations and 93% in terms of storage. For the Borehole function, we have $r_\ell = R_\ell^2$, i.e. we can not compress the FTT approximation further. In this case, the direct TT approximation is slightly more efficient. At the same time, this demonstrates that all other test functions, taken from a wide range of applications, can be stored more efficiently using our approach. Details can be found in Table 4.1.

Genz functions In order to quantify the impact of the dimension on the approximation error, we study the Genz [136] functions defined in Appendix 4.A.2. Our numerical results for the approximation using a fixed polynomial degree, displayed in Figure 4.2, demonstrate that Algorithm 10 requires fewer function evaluation compared to a direct TT-cross approximation. However, in very large dimensions ($d > 300$) the direct TT-cross approach leads to slightly more accurate approximations. This might be caused by the fact that, for Algorithm 10, the error of d projections needs to be taken into account in addition to the TT-cross approximation error. Moreover, approximating the smaller tensor $\mathcal{C}$ instead of $\mathcal{T}$ directly, might lead to slightly worse conditioned inverse matrices

Function	Algorithm	Error	# evals	# dofs	$\max_\ell R_\ell$	$\max_\ell r_\ell$
Ackley	EFTT	1.84e-02	63168	15965	15	10
	DirectTT	1.84e-02	570176	226555	18	
Alpine	EFTT	5.80e-03	4677	1448	2	2
	DirectTT	5.80e-03	6860	2400	2	
Dixon	EFTT	5.72e-14	11879	3549	3	5
	DirectTT	2.66e-14	13022	5100	3	
Exponential	EFTT	2.10e-14	2108	707	1	1
	DirectTT	2.09e-14	2585	700	1	
Griewank	EFTT	1.69e-07	8091	2252	3	3
	DirectTT	1.54e-07	13022	5100	3	
Michalewicz	EFTT	4.05e-02	4677	1448	2	2
	DirectTT	4.05e-02	6860	2400	2	
Piston	EFTT	3.22e-09	202876	73978	24	11
	DirectTT	3.02e-09	991669	412186	18	
Qing	EFTT	1.11e-13	5482	2172	2	3
	DirectTT	2.29e-14	6860	2400	2	
Rastrigin	EFTT	2.29e-14	4677	1448	2	2
	DirectTT	2.30e-14	6860	2400	2	
Rosenbrock	EFTT	3.02e-14	10970	2798	3	4
	DirectTT	2.64e-14	13022	5100	3	
Schaffer	EFTT	6.76e-02	1063304	288006	39	40
	DirectTT	6.73e-02	1510513	766565	30	
Schwefel	EFTT	6.58e-04	4677	1448	2	2
	DirectTT	6.58e-04	6860	2400	2	
Borehole	EFTT	3.95e-02	14365	3270	2	4
	DirectTT	3.95e-02	10178	2336	2	
OTL Circuit	EFTT	3.85e-11	16064	3277	5	5
	DirectTT	8.31e-12	27764	8300	4	
Robot Arm	EFTT	6.65e-02	523301	113365	35	36
	DirectTT	6.88e-02	752815	392234	32	
Wing Weight	EFTT	3.72e-14	6692	2072	2	2
	DirectTT	8.11e-14	10439	3600	2	
Friedman	EFTT	5.49e-10	12312	2376	4	4
	DirectTT	1.04e-11	14630	3136	3	
G & L	EFTT	2.52e-05	3278	1034	2	2
	DirectTT	2.52e-05	6651	1800	2	
G & P 8D	EFTT	3.07e-11	39588	8121	7	7
	DirectTT	2.68e-11	74693	30160	5	
D & P Exp	EFTT	1.56e-14	1990	616	2	2
	DirectTT	1.55e-14	2087	800	2	

Table 4.1 – We apply Algorithm 10 (EFTT) and an approximation of the evaluation tensor using Algorithm 3 (DirectTT) to approximate the test functions defined in Appendix 4.A.1. For each function, we display the estimated L^2 error of the approximations, the number of function evaluations required to construct the approximation, the degrees of freedom in the approximation and the largest R_ℓ and r_ℓ of the resulting approximation. The numbers displayed are the mean (geometric mean for the error and arithmetic mean for all other quantities) after approximating each function 100 times.

in (2.8).

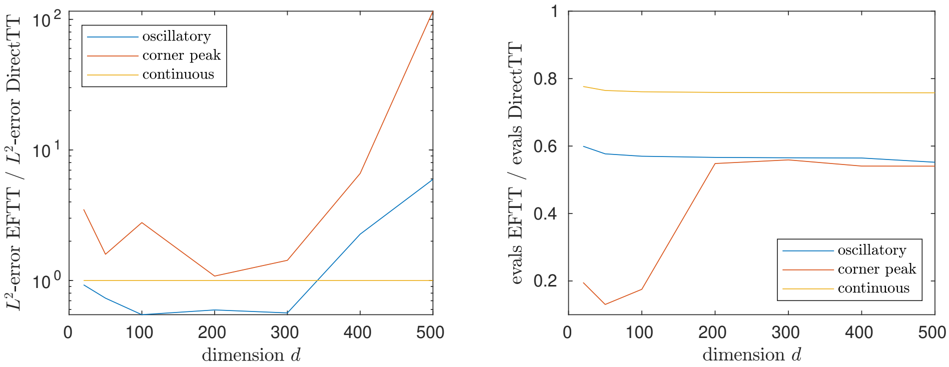


Figure 4.2 – We apply Algorithm 10 (EFTT) and an approximation of the evaluation tensor using Algorithm 3 (DirectTT) to approximate the Genz functions (see Appendix 4.A.2) with fixed polynomial degree for dimension $d \in \{20, 50, 100, 200, 300, 400, 500\}$. Left: We plot the ratio of L^2 error for EFFT / DirectTT, i.e. for the function 'corner peak' with $d = 20$ the EFTT approximation is around halve an order less accurate than the DirectTT approximation. Right: We plot the ratio of the function evaluations required to compute the approximation for EFFT / DirectTT, i.e. for the function 'corner peak' with $d = 20$ EFTT we use only around 20% of the number of evaluations required for DirectTT. Throughout this figure, we sample 30 different parameters for the Genz functions as in [41] and display the geometric mean of the error and the arithmetic mean for the number of function evaluations.

4.3.2 Comparison to the FTT approximation algorithm

In this section, we compare the performance of our proposed Algorithm 10 based on the EFTT format (4.1) to approximation algorithm proposed in [139, 141] and implemented in the c3py package [‡]. The c3py algorithm uses a continuous variant of the TT-cross algorithm to compute approximations in the FTT format (2.12).

The c3py package is based on the Legendre polynomials $P_0(x) = 1,\ P_1(x) = x,\ (k+2)P_{k+2}(x) = (2k+3)xP_{k+1}(x) - (k+1)P_k(x)$ for $k = 0, \dots, m-2$ [197] with degree at most $m \in \mathbb{N}$ instead of Chebyshev polynomials. This leads to multivariate polynomial approximations with degree at most $(m_1, \dots, m_d)$ of the form

$$\sum_{i_1=0}^{m_1} \cdots \sum_{i_d=0}^{m_d} \mathcal{B}_{i_1,\dots,i_d} P_{i_1}(x_1) \cdots P_{i_d}(x_d), \tag{4.7}$$

where $\mathcal{B} \in \mathbb{R}^{(m_1+1)\times\cdots\times(m_d+1)}$. Since the asymptotic convergence rates of Legendre and Chebyshev interpolation are different [46], a fair comparison requires that we adjust our approximation algorithm to also return an approximation of the form (4.7). An approximation of the form (4.7) can be obtained from the evaluation tensor $\mathcal{T} \in$

[‡] The c3py package is available from https://github.com/goroda/Compressed-Continuous-Computation

$\mathbb{R}^{(n_1+1)\times\cdots\times(n_d+1)}$ (2.18) as $\mathcal{B} = \mathcal{T} \times_1 E^{(1)} \times_2 \cdots \times_d E^{(d)}$, where $E^{(\ell)} \in \mathbb{R}^{(n_\ell+1)\times(m_\ell+1)}$ is defined as

$$E^{(\ell)} = \begin{pmatrix} \frac{1}{2} w_0^{(\ell)} P_0(x_0^{(\ell)}) & \frac{1}{2} w_1^{(\ell)} P_0(x_1^{(\ell)}) & \dots & \frac{1}{2} w_{m_\ell}^{(\ell)} P_0(x_{m_\ell}^{(\ell)}) \\ \frac{3}{2} w_0^{(\ell)} P_1(x_0^{(\ell)}) & \frac{3}{2} w_1^{(\ell)} P_1(x_1^{(\ell)}) & \dots & \frac{3}{2} w_{m_\ell}^{(\ell)} P_1(x_{m_\ell}^{(\ell)}) \\ \frac{5}{2} w_0^{(\ell)} P_2(x_0^{(\ell)}) & \frac{5}{2} w_1^{(\ell)} P_2(x_1^{(\ell)}) & \dots & \frac{5}{2} w_{m_\ell}^{(\ell)} P_2(x_{m_\ell}^{(\ell)}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2n_\ell+1}{2} w_0^{(\ell)} P_{n_\ell}(x_0^{(\ell)}) & \frac{2n_\ell+1}{2} w_1^{(\ell)} P_{n_\ell}(x_1^{(\ell)}) & \dots & \frac{2n_\ell+1}{2} w_{m_\ell}^{(\ell)} P_{n_\ell}(x_{m_\ell}^{(\ell)}) \end{pmatrix},$$

with nonnegative weights $w_{i_\ell}^{(\ell)}$ defined as in Remark 2.12. The matrix $E^{(\ell)}$ encodes Clenshaw-Curtis quadrature [74] in $n_\ell + 1$ nodes to (approximately) compute the Legendre coefficients via L^2-projections. Using analogous constructions as in Section 7.1, we can transform an extended TT approximation of the evaluation tensor (4.4) into an EFTT approximation with univariate functions represented in terms of linear combinations of Legendre polynomials.

Remark 4.3. *Gauss-Legendre quadrature in $n+1$ points integrates a polynomials of degree $2n+1$ exactly, whereas Clenshaw-Curtis quadrature is only exact for polynomials of degree n. We could define the evaluation tensor $\mathcal{T}$ and the matrices E^ℓ differently to encode Gauss-Legendre quadrature instead of Clenshaw-Curtis quadrature. However, in practice, both quadrature rules typically perform well as pointed out in [325].*

In the following experiments we set $n_\ell = 2m_\ell$ for $\ell = 1, \dots, d$ to ensure accurate quadrature. To determine the polynomial degrees m_ℓ adaptively (see Remark 4.2), we follow the fiber adaptation strategy of c3py (see [139, Section 3.6.1]): We progressively increase the degrees until four sequential Legendre coefficients are smaller than the tolerance 10^{-10} or until the maximum of $m_\ell = 105$ has been reached. In the c3py algorithm, we set all tolerances to 10^{-10}.

Benchmark functions. We apply both the c3py algorithm from [141] and our novel Algorithm 10 to approximate the benchmark functions defined in Appendix 4.A.1. Our numerical results displayed in Table 4.2 demonstrate that our novel algorithm typically requires fewer function evaluations and still achieves the same accuracy compared to c3py. For the Ackley function, our approach reduces the number of required function evaluations by more than 96% and the storage by more than 90%. At the same time, our approach is slightly more accurate.

Integration of sin function In the following, we repeat the experiment from [139, Figure 3-6]. The function $f(x_1, \dots, x_d) = \sin(x_1 + x_2 + \cdots + x_d)$ can be represented in FTT format with TT ranks $(2, 2, \dots, 2)$ and its integral over the domain $[0, 1]^d$ is known analytically [139]. In Figure 4.3, we plot the error of the integral of the approximations

Function	Algorithm	Error	# evals	# dofs	$\max_\ell m_\ell$	$\max_\ell R_\ell$	$\max_\ell r_\ell$
Ackley	EFTT	1.22e-02	67107	18465	105	15	9
	c3py	1.81e-02	2232528	197760	105	16	
Alpine	EFTT	4.08e-03	5464	1518	105	2	2
	c3py	6.43e-03	50656	2520	105	2	
Dixon	EFTT	3.21e-14	13756	3714	105	3	5
	c3py	2.89e-14	13752	435	21	3	
Exponential	EFTT	4.82e-15	946	196	27	1	1
	c3py	1.71e-14	10518	133	19	1	
Griewank	EFTT	8.21e-08	9139	2358	105	3	3
	c3py	3.52e-06	51466	4459	105	3	
Michalewicz	EFTT	2.54e-02	5464	1518	105	2	2
	c3py	1.45e-01	48745	2443	105	2	
Piston	EFTT	3.71e-09	174188	69019	33	23	11
	c3py	3.85e-05	251760	66080	35	24	
Qing	EFTT	5.54e-13	6996	2277	105	2	3
	c3py	2.86e-14	11776	136	21	2	
Rastrigin	EFTT	1.91e-14	5463	1518	105	2	2
	c3py	1.86e-10	22288	1342	63	2	
Rosenbrock	EFTT	1.25e-14	4106	835	27	3	4
	c3py	9.43e-14	11530	633	14	3	
Schaffer	EFTT	7.19e-02	173787	35016	105	16	17
	c3py	1.22e-01	3465760	214200	105	20	
Schwefel	EFTT	4.00e-04	5463	1518	105	2	2
	c3py	5.45e-04	50656	2496	104	2	
Borehole	EFTT	3.95e-02	6552	1116	32	2	4
	c3py	2.08e-03	14346	577	70	2	
OTL Circuit	EFTT	7.93e-11	6670	1083	27	5	5
	c3py	4.07e-08	15674	1782	28	5	
Robot Arm	EFTT	8.12e-02	499954	54760	94	12	27
	c3py	3.85e-01	2018017	228439	105	20	
Wing Weight	EFTT	2.83e-14	2867	560	24	2	2
	c3py	2.15e-13	12224	554	19	2	
Friedman	EFTT	2.16e-11	5238	404	19	3	4
	c3py	8.08e-05	12142	710	15	4	
G & L	EFTT	4.95e-06	1547	356	29	2	2
	c3py	3.51e-02	13928	374	105	2	
G & P 8D	EFTT	4.77e-11	19527	3902	24	6	7
	c3py	9.54e-10	27336	5136	21	7	
D & P Exp	EFTT	1.13e-14	2404	646	105	2	2
	c3py	4.78e-10	12162	336	49	2	

Table 4.2 – We apply Algorithm 10 (EFTT) and the algorithm in the c3py package to approximate the test functions defined in Appendix 4.A.1 by functional low-rank approximation with Legendre polynomial basis functions (4.7). For each function, we display the estimated L^2-error of the approximations, the number of function evaluations required to construct the approximation, the degrees of freedom in the approximation and the largest m_ℓ, R_ℓ and r_ℓ of the resulting approximation.

computed via Algorithm 10 and the c3py algorithm. The figure shows, that our novel approach achieves a similar level of accuracy. The difference in the number of function evaluations is rather small, since the multilinear ranks and TT ranks of the function are small.

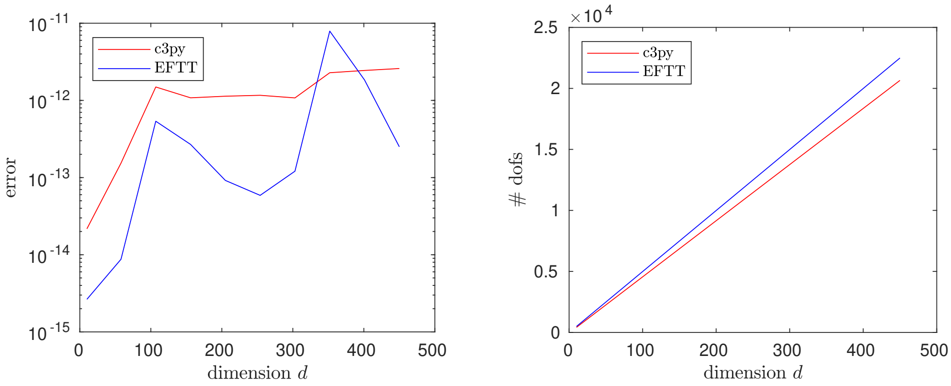


Figure 4.3 – We apply Algorithm 10 (EFTT) and the algorithm in the c3py package to approximate $f(x_1, \ldots, x_d) = \sin(x_1 + x_2 + \cdots + x_d)$ by functional low-rank approximation with Legendre polynomial basis functions (4.7) as in [139, Figure 3-6]. Left: We plot the relative error of the integral of the approximations. Right: We plot the number of function evaluations required to compute the approximation.

Application: Uncertainty quantification. A classical application of multivariate function approximation is the computation of surrogates for uncertainty quantification [308, 341]. For the approximation of the quantity of interest mapping defined in Appendix 4.A.3, we find in Table 4.3 that Algorithm 10 leads to approximations requiring less storage compared to c3py.

		error	# evals	# dofs
$d = 4$	c3py	$4.21 \cdot 10^{-5}$	360	168
	EFTT	$1.05 \cdot 10^{-3}$	337	60
$d = 9$	c3py	$9.79 \cdot 10^{-5}$	960	448
	EFTT	$1.30 \cdot 10^{-3}$	757	153
$d = 16$	c3py	$1.73 \cdot 10^{-4}$	1800	840
	EFTT	$7.62 \cdot 10^{-4}$	1345	240

Table 4.3 – We apply Algorithm 10 (EFTT) and the algorithm in the c3py package to approximate the quantity of interest map $Q : [-1, 1]^d \to \mathbb{R}$ defined in Appendix 4.A.3 for $d \in \{4, 9, 16\}$. All tolerances are set to 10^{-3}. The table displays the L^2-error, the required number of evaluations (evals) and the number of degrees of freedom (dofs) for both approximations.

4.A Appendix

4.A.1 Benchmark functions

In Table 4.4 and Table 4.5, we define the benchmark functions for our numerical experiments. Note that the functions are defined on different tensor product domains. In our experiments, we map the domain of these functions onto $[-1,1]^d$ using an affine linear transformation.

Function	d	Domain	References
$f_{\text{Ackley}}(\mathbf{x}) = -20\exp\Big(-0.2\sqrt{\frac{1}{7}\sum_{i=1}^{7}(x_i)^2}\Big) - \exp\Big(\frac{1}{7}\sum_{i=1}^{7}\cos(2\pi(x_i))\Big) + 20 + e^1$	7	$[-32.768, 32.768]^7$	[177, 310]
$f_{\text{Alpine}}(\mathbf{x}) = \sum_{i=1}^{7} \lvert x_i \sin(x_i) + 0.1x_1 \rvert$	7	$[-10, 10]^7$	[177, 269]
$f_{\text{Dixon}}(\mathbf{x}) = (x_1 - 1)^2 + \sum_{i=2}^{7} i \cdot (2x_i^2 - x_{i-1})^2$	7	$[-10, 10]^7$	[177, 310]
$f_{\text{Exponential}}(\mathbf{x}) = -\exp\Big(-\frac{1}{2}\sum_{i=1}^{7} x_i^2\Big)$	7	$[-1, 1]^7$	[177, 268]
$f_{\text{Griewank}}(\mathbf{x}) = \sum_{i=1}^{7} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\Big(\frac{x_i}{\sqrt{i}}\Big) + 1$	7	$[-600, 600]^7$	[177, 310]
$f_{\text{Michalewicz}}(\mathbf{x}) = -\sum_{i=1}^{7} \sin\Big(x_i\Big) \sin^{20}\Big(\frac{ix_i^2}{\pi}\Big)$	7	$[0, \pi]^7$	[310, 329]
$f_{\text{Piston}}(\mathbf{x})$	7	see text	[310, 344]
$f_{\text{Qing}}(\mathbf{x}) = \sum_{i=1}^{7}(x_i^2 - i)^2$	7	$[0, 500]^7$	[177, 265]
$f_{\text{Rastrigin}}(\mathbf{x}) = 70 + \sum_{i=1}^{7}(x_i^2 - 10\cos(2\pi \cdot x_i))$	7	$[-5.12, 5.12]^7$	[98, 310]
$f_{\text{Rosenbrock}}(\mathbf{x}) = \sum_{i=1}^{6}(100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$	7	$[-2.048, 2.048]^7$	[177, 310]
$f_{\text{Schaffer}}(\mathbf{x}) = \sum_{i=1}^{6}\Bigg(0.5 + \frac{\sin^2\Big(\sqrt{x_i^2 + x_{i+1}^2}\Big) - 0.5}{\Big(1 + 0.001(x_i^2 + x_{i+1}^2)\Big)^2}\Bigg)$	7	$[-100, 100]^7$	[177, 310]
$f_{\text{Schwefel}}(\mathbf{x}) = 2932.8803 - \sum_{i=1}^{7} x_i \cdot \sin(\sqrt{\lvert x_i \rvert})$	7	$[-500, 500]^7$	[98, 310]

Table 4.4 – Test functions from [71, Table 1].

Function	d	Domain	References
$f_{\text{Borehole}}(\mathbf{x})$	8	see text	[12, 310]
$f_{\text{OTL Circuit}}(\mathbf{x})$	6	see text	[233, 310]
$f_{\text{Robot Arm}}(\mathbf{x})$	8	see text	[12, 310]
$f_{\text{Wing Weight}}(\mathbf{x})$	10	see text	[123, 310]
$f_{\text{Friedman}}(\mathbf{x}) = 10\sin(\pi x_1 x_2) + 20\,(x_3 - 0.5)^2 + 10x_4 + 5x_5$	5	$[0,1]^5$	[128, 310]
$f_{\text{G\&L}}(\mathbf{x}) = \exp\left[\sin\left(\left(0.9\,(x_1 + 0.48)\right)^{10}\right)\right] + x_2 x_3 + x_4$	6	$[0,1]^6$	[144, 310]
$f_{\text{D\&P 8D}}(\mathbf{x}) = 4\left(x_1 - 2 + 8x_2 - 8x_2^2\right)^2 + (3 - 4x_2)^2 + 16\sqrt{x_3+1}\,(2x_3 - 1)^2 + \sum_{i=4}^{8} i \ln\left(1 + \sum_{j=3}^{i} x_j\right)$	8	$[0,1]^8$	[94, 310]
$f_{\text{D\&P Exp}}(\mathbf{x}) = 100\left(e^{-2/x_1^{1.75}} + e^{-2/x_2^{1.5}} + e^{-2/x_3^{1.25}}\right)$	3	$[0,1]^3$	[94, 310]

Table 4.5 – Test functions from [139, Table 3.2].

Some of these functions do not fit into the format of the table. These are defined in the following:

$$f_{\text{Piston}}(M, S, V_0, k, P_0, T_a, T_0) = 2\pi\sqrt{\frac{M}{k + S^2 \frac{P_0 V_0}{T_0} \frac{T_a}{V^2}}},$$

where

$$V = \frac{S}{2k}\left(\sqrt{A^2 + 4k\frac{P_0 V_0}{T_0} T_a} - A\right) \text{ and } A = P_0 S + 19.62M - \frac{kV_0}{S},$$

with $M \in [30, 60]$, $S \in [0.005, 0.02]$, $V_0 \in [0.002, 0.01]$, $k = [1000, 5000]$, $P_0 \in [90000, 110000]$, $T_a \in [290, 296]$, $T_0 \in [340, 360]$.

$$f_{\text{Borehole}}(r_w, r, T_u, H_u, T_l, H_l, L, K_w) = \frac{2\pi T_u (H_u - H_l)}{\log(r/r_w)\left(1 + \frac{2LT_u}{\log(r/r_w) r_w^2 K_w} + \frac{T_u}{T_l}\right)},$$

with $r_w \in [0.05, 0.15]$, $r \in [100, 50000]$, $T_u \in [63070, 115600]$, $H_u \in [990, 1110]$, $T_l \in [63.1, 116]$, $H_l \in [700, 820]$, $L \in [1120, 1680]$, $K_w \in [9855, 12045]$.

$$f_{\text{OTL Circuit}}(b_1, b_2, f, c_1, c_2, \beta) = \frac{(\frac{12b_2}{b_1+b_2} + 0.74)\beta(c_2 + 9)}{\beta(c_2 + 9) + f} + \frac{11.35f}{\beta(c_2 + 9) + f} + \frac{0.74f\beta(c_2 + 9)}{(\beta(c_2 + 9) + f)c_1},$$

with $b_1 \in [50, 150]$, $b_2 \in [25, 70]$, $f \in [0.5, 3]$, $c_1 \in [1.2, 2.5]$, $c_2 \in [0.25, 1.2]$, $\beta \in [50, 300]$.

$$f_{\text{Robot Arm}}(\theta_1, \theta_2, \theta_3, \theta_4, L_1, L_2, L_3, L_4) = \sqrt{u^2 + v^2},$$

where $u = \sum_{i=1}^4 L_i \cos(\sum_{j=1}^4 \theta_i)$, $v = \sum_{i=1}^4 L_i \sin(\sum_{j=1}^4 \theta_i)$ and $\theta_i \in [0, 2\pi]$, $L_i \in [0, 1]$ for $i = 1, \ldots, 4$.

$$\begin{aligned} &f_{\text{Wing Weight}}(S_w, W_f, A, \Delta, q, \lambda, t_c, N_z, W_d, W_p) = \\ &\quad 0.036 S_w^{0.758} W_f^{0.0035} \Big(\frac{A}{\cos^2(\Delta)}\Big)^{0.6} q^{0.006} \lambda^{0.04} \Big(\frac{100 t_c}{\cos(\Delta)}\Big)^{-0.3} (N_z W_d)^{0.49} + S_w W_p, \end{aligned}$$

with $S_w \in [150, 200], W_f \in [220, 300]$, $A \in [6, 10]$, $\Delta \in [-10, 10]$, $q \in [16, 45]$, $\lambda \in [0.5, 1]$, $t_c \in [0.08, 0.18]$, $N_z \in [2.5, 6]$, $W_d \in [1700, 2500]$, $W_p \in [0.025, 0.08]$.

4.A.2 Genz functions

In the following, we define the Genz functions [136], which are frequently used to evaluate function approximation and integration schemes. On the domain $[-1, 1]^d$, we consider

$$\begin{aligned} f_1(\mathbf{x}) &= \cos\left(2\pi w_1 + \sum_{i=1}^d c_i \frac{x_i + 1}{2}\right) && \text{(oscillatory)} \\ f_2(\mathbf{x}) &= \left(1 + \sum_{i=1}^d c_i \frac{x_i + 1}{2}\right)^{-(d+1)} && \text{(corner peak)} \\ f_3(\mathbf{x}) &= \exp\left(-\sum_{i=1}^d c_i^2 \left|\frac{x_i + 1}{2} - w_i\right|\right) && \text{(continuous)} \end{aligned}$$

The parameters w_i and c_i are drawn uniformly from $[0, 1]$, where w_i act as a shift for the functions while c_i determines the approximation difficulty of the functions. We normalized c_i such that

$$\sum_{i=1}^d |c_i| = \frac{b}{d^h},$$

where the scaling constants h and b are defined for each function as in [41, Table 1]. Note that f_1 can be represented in FTT format (2.12) with $\max R_\ell = 2$. The function f_3 is separable. For f_2, we are not aware of any analytic FTT representation.

4.A.3 Parametric PDE problem

In the following, we recall the example from [207, Section 4]. Assume $\sqrt{d} \in \mathbb{N}$. Let $\Omega = [0, 1]^2$. We consider the parametric elliptic PDE

$$-\nabla \cdot (a(x, p) \nabla u(x, p)) = 1, \quad x \in \Omega \tag{4.8}$$

with homogeneous Dirichlet boundary conditions and parameter $p \in [-1,1]^d$. We define the piecewise constant coefficient $a(x,p) : \Omega \times \mathbb{R}^d$ as

$$a(x,p) = \begin{cases} 1.25 + 0.75 p_{\sqrt{d}(t-1)+s} & x \in \Omega_{s,t} \\ 1 & \text{otherwise,} \end{cases}$$

where we denote the disk with radius $\rho = 1/(4\sqrt{d}+2)$ centered around $(\rho(4s-1), \rho(4t-1))$ by $\Omega_{s,t}$ for $s,t = 1,\dots,\sqrt{d}$. In our numerical experiments, we approximate the quantity of interest $Q : [-1,1]^d \to \mathbb{R}$ defined as

$$Q(p) = \int_0^1 \int_0^1 u(x,p) \partial x_1 \partial x_2,$$

where $u(x,p)$ denotes the solution of the PDE (4.8) for the given parameter $p \in [-1,1]^d$. For each value of p, we solve the resulting PDE using a discretization based on linear finite elements [48].

5 Multi-marginal optimal transport

In this chapter, we study the solution of multi-marginal optimal transport problems, i.e. we search the nonnegative transport plan tensor $\mathcal{P}$ of order d, which minimizes $\langle \mathcal{C}, \mathcal{P} \rangle$ subject to marginal constraints, where $\mathcal{C}$ denotes a given nonnegative cost tensor of order d. After entropic regularization [82] the problem is equivalent to a tensor scaling problem of $\exp(-\mathcal{C}/\eta)$ [37] and can be solved using the Sinkhorn algorithm [58]. The multi-marginal Sinkhorn algorithm crucially relies on the repeated evaluation of marginals of the rescaled tensor $\exp(-\mathcal{C}/\eta)$.

Using a novel approach, we combine the ideas of exploiting underlying graphical models [150] and using low-rank approximations [7, 10] to compute marginals more efficiently. When the structure of transport plans is specified by a graphical model, we observe that the dual of this model is a tensor network [273], which contains a tensor network representation of $\exp(-\mathcal{C}/\eta)$. At the same time low-rank approximations of $\exp(-\mathcal{C}/\eta)$ can also be represented as tensor networks [247]. Facilitating this point of view, marginals can in both cases be computed by contracting [272] the tensor network and the scaling parameters. Note that the belief propagation and the junction tree algorithm for graphical models correspond to a particular order of contracting this network [273]. For tensor networks derived from graphical models, we propose to potentially accelerate the computation of marginals further by replacing tensors in the network by low-rank approximations. This yields a modified tensor network and can be seen as an approximation of $\exp(-\mathcal{C}/\eta)$. In our numerical experiments, we provide an example illustrating that our approach to introduce low-rank approximations in the tensor network is more efficient than directly working with the graphical model and more accurate than a direct tensor train approximation [249] of $\exp(-\mathcal{C}/\eta)$. We also demonstrate that our approach offers the potential to greatly speed up the computation of color transfer between several images without altering the resulting image significantly.

Moreover, we provide theoretical bounds on the error caused by introducing such approximations. In Theorem 5.2, we state a bound for the impact of using an approximation of

$\exp(-\mathcal{C}/\eta)$ on the entropically regularized transport cost. This result is a generalization of the bound in [10] for classical optimal transport problems. In contrast to the asymptotic bound in [7], our result contains explicit constants and does not assume that $\mathcal{C}$ is low-rank. We also would like to point out that the bound in [7] relies on the fact the rank of $\exp(-\mathcal{C}/\eta)$ can be bounded based on the rank of $\mathcal{C}$. The practical usefulness of these results is impeded by the fact that the elementwise exponential tends to increase (approximate) ranks drastically. For example, to obtain a reasonably good approximation of the elementwise exponential of a random 1000×1000 rank-5 matrix by truncating all singular values smaller than 10^{-10} one easily ends up with a matrix of rank 800 or larger. In Theorem 5.3, we state how the parameters and tolerances need to be selected to obtain an accurate approximation of the original problem without regularization. This generalizes previous results in [217] from the case of using the tensor $\exp(-\mathcal{C}/\eta)$ directly to our case of using an approximation instead. In Lemma 5.2, we link approximations of parts in the tensor network to approximations of $\exp(-\mathcal{C}/\eta)$.

This chapter is based on the article [305]. Its remainder is structured as follows. In Section 5.1, we define the multi-marginal optimal transport problem and summarize the convergence results for the multi-marginal Sinkhorn algorithm in [127, 217]. The impact of approximating $\exp(-\mathcal{C}/\eta)$ is analyzed in Section 5.2. In Section 5.3, we derive the tensor network structure of $\exp(-\mathcal{C}/\eta)$ for transport plans represented by graphical models. Approximations of parts of this network are connected to approximations of $\exp(-\mathcal{C}/\eta)$ in Section 5.4. Our numerical experiments in Section 5.5 demonstrate how the multi-marginal Sinkhorn algorithm can be accelerated by introducing low-rank approximations into the tensor network representation of $\exp(-\mathcal{C}/\eta)$.

5.1 Multi-marginal optimal transport and the Sinkhorn algorithm

Throughout this chapter, we let $\Delta^n = \{x \in \mathbb{R}^n_{>0} \colon \|x\|_1 = 1\}$ denote the set of strictly positive probability vectors of length n, where $\mathbb{R}_{>0}$ denotes the strictly positive real numbers. Further, we let $\mathbb{R}_+$ denote the nonnegative real numbers. For a tensor $\mathcal{X} \in \mathbb{R}_+^{n_1 \times \cdots \times n_d}$ containing the joint probability distribution of d discrete random variables, we denote the ℓth marginal (distribution) in Δ^{n_ℓ} by

$$r_\ell(\mathcal{X}) = \mathsf{vec}(\mathcal{X} \times_1 \mathbf{1}_{n_1}^T \cdots \times_{\ell-1} \mathbf{1}_{n_{\ell-1}}^T \times_{\ell+1} \mathbf{1}_{n_{\ell+1}}^T \cdots \times_d \mathbf{1}_{n_d}^T),$$

where $\mathbf{1}_n \in \mathbb{R}^n$ denotes the vector of all ones. Throughout this chapter, the operations log and exp are always applied elementwise.

5.1.1 Mathematical setting

Given $d \geq 2$ marginals $r_\ell \in \Delta^{n_\ell},\ \ell = 1, \ldots, d$ and a cost tensor $\mathcal{C} \in \mathbb{R}_+^{n_1 \times n_2 \times \cdots \times n_d}$, the discrete multi-marginal optimal transport problem [238] is given by

$$\min_{\mathcal{P} \in B(r_1,\ldots,r_d)} \langle \mathcal{P}, \mathcal{C} \rangle = \min_{\mathcal{P} \in B(r_1,\ldots,r_d)} \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \mathcal{P}_{i_1,\ldots,i_d} \mathcal{C}_{i_1,\ldots,i_d}, \tag{5.1}$$

where the set of feasible transport plans is given by

$$B(r_1, \ldots, r_d) = \left\{ \mathcal{P} \in \mathbb{R}_+^{n_1 \times \cdots \times n_d} | r_\ell(\mathcal{P}) = r_\ell \text{ for } \ell = 1, \ldots, d \right\}.$$

Note that (5.1) is a linear optimization problem with $n_1 \cdot n_2 \cdots n_d$ degrees of freedom.

To solve (5.1) efficiently, it is common to add entropic regularization [82]. In the multi-marginal setting, the entropy takes the form

$$H(\mathcal{P}) = -\langle \mathcal{P}, \log(\mathcal{P}) \rangle.$$

Given a regularization parameter $\eta > 0$, the regularized problem takes the form

$$\min_{\mathcal{P} \in B(r_1,\ldots,r_d)} V_{\mathcal{C}}^{\eta}(\mathcal{P}), \tag{5.2}$$

where

$$V_{\mathcal{C}}^{\eta}(\mathcal{P}) := \langle \mathcal{P}, \mathcal{C} \rangle - \eta H(\mathcal{P})$$

is called entropic transport cost. It is known that the regularized problem (5.2) has a unique minimizer [37], which takes the form

$$\mathcal{P}_\eta^* = \mathcal{K} \times_1 \operatorname{diag}(\exp(\beta_1)) \cdots \times_d \operatorname{diag}(\exp(\beta_d)), \tag{5.3}$$

where $\mathcal{K} := \exp(-\mathcal{C}/\eta)$ is called Gibbs kernel [58] and $\operatorname{diag}(\beta_\ell)$ denotes the diagonal matrix containing the entries of the so-called scaling parameters $\beta_\ell \in \mathbb{R}^{n_\ell}$ on its diagonal. The solution of the regularized problem (5.2) converges to a solution of (5.1) when $\eta \to 0$; see [35, 211].

5.1.2 Multi-marginal Sinkhorn algorithm

The multi-marginal Sinkhorn algorithm for solving (5.2) proceeds by iteratively updating the scaling parameters in (5.3). Let

$$\mathcal{P}_\eta^{(t)} = \mathcal{K} \times_1 \operatorname{diag}(\exp(\beta_1^{(t)})) \cdots \times_d \operatorname{diag}(\exp(\beta_d^{(t)})) \tag{5.4}$$

denote the scaled tensor obtained after t iterations. In each iteration one selects an index ℓ and updates the ℓth scaling parameter via $\beta_\ell^{(t+1)} = \log(r_\ell) - \log(r_\ell(\mathcal{P}_\eta^{(t)})) + \beta_\ell^{(t)}$, while the other vectors remain unchanged. For the choice of indices it has been suggested to traverse them cyclically [37] or use a greedy heuristics [127, 217]. The multi-marginal Sinkhorn algorithm is terminated once the stopping criterion

$$\sum_{\ell=1}^{d} \|r_\ell(\mathcal{P}_\eta^{(t)}) - r_\ell\|_1 \leq \varepsilon_{\mathsf{stop}} \tag{5.5}$$

is satisfied, where $\varepsilon_{\mathsf{stop}} > 0$ denotes a prescribed tolerance. Algorithm 11 summarizes the described procedure.

Algorithm 11 Multi-marginal Sinkhorn algorithm

```
1: Input: cost tensor $\mathcal{C}$, marginals $r_1, \dots, r_d$, regularization parameter $\eta$
2: Output: transport plan $\mathcal{P}_\eta^{(t)}$
3: $\beta_\ell^{(0)} = \mathbf{0} \in \mathbb{R}^{n_i}$ for $\ell = 1, \dots, d$ and $\mathcal{K} = \exp(-\mathcal{C}/\eta)$
4: for $t = 0, 1, \dots$ until stopping criterion (5.5) is satisfied
5:     $\mathcal{P}_\eta^{(t)} = \mathcal{K} \times_1 \mathrm{diag}(\exp(\beta_1^{(t)})) \cdots \times_d \mathrm{diag}(\exp(\beta_d^{(t)}))$
6:     Let $\ell_{\mathsf{next}}$ denote the index of the scaling parameter that should be updated next.
7:     $\beta_\ell^{(t+1)} = \begin{cases} \log(r_\ell) - \log(r_\ell(\mathcal{P}_\eta^{(t)})) + \beta_\ell^{(t)} & \ell = \ell_{\mathsf{next}} \\ \beta_\ell^{(t)} & \ell \neq \ell_{\mathsf{next}} \end{cases}$
```

When using cyclic order, it follows from an interpretation as Bregman projections [37] that the multi-marginal Sinkhorn algorithm converges. For greedy strategies, Theorem 5.1 below summarizes the statements of [217, Theorem 4.3] and [127, Theorem 3.4], which bound the number of iterations until the stopping criterion is reached. Note that the bound in a) gives a better rate with respect to $\varepsilon_{\mathsf{stop}}$, but the bound depends on r_ℓ, whereas the bound in b) does not involve r_ℓ.

Theorem 5.1. *Let $\mathcal{C} \in \mathbb{R}_+^{n_1 \times \cdots \times n_d}$, $r_\ell \in \Delta^{n_\ell}$ for $\ell = 1, \dots, d$, $0 < \eta < \frac{1}{2}$, and $\varepsilon_{\mathsf{stop}} > 0$.*

a) Suppose that Algorithm 11 selects the index of the scaling parameter in iteration t according to

$$\ell_{\mathsf{next}} = \operatorname*{arg\,max}_{\ell \in \{1,\dots,d\}} \mathbf{1}^T (r_\ell(\mathcal{P}_\eta^{(t)}) - r_\ell) + \mathbf{1}^T \Big(r_\ell * \log \Big(r_\ell(\mathcal{P}_\eta^{(t)}) * r_\ell^{-1} \Big) \Big),$$

where $*$ denotes the elementwise product and r_ℓ^{-1} denotes the elementwise inverse. Then the number of iterations to reach the stopping criterion (5.5) is bounded by

$$t \leq 2 + 2d^2 \varepsilon_{\mathsf{stop}}^{-1} \Big(\eta^{-1} \|\mathcal{C}\|_\infty - \log \min_{1 \leq \ell \leq d} \min_{1 \leq i \leq n_\ell} (r_\ell)_i \Big). \tag{5.6}$$

b) Assume that $n = n_1 = \cdots = n_d$ and suppose that Algorithm 11 normalizes $\mathcal{P}_\eta^{(0)}$ to *have* ℓ^1*-norm* 1 *and selects the index of the scaling parameter in iteration* t *according to*

$$\ell_{\mathit{next}} = \operatorname*{arg\,max}_{\ell \in \{1,\dots,d\}} \left\| r_\ell(\mathcal{P}_\eta^{(t)}) - \frac{\langle r_\ell, r_\ell(\mathcal{P}_\eta^{(t)})\rangle}{\|r_\ell\|_2^2} r_\ell \right\|_1 . \tag{5.7}$$

Then the number of iterations needed to reach the stopping criterion

$$\max_{\ell \in \{1,\dots,d\}} \left\| r_\ell(\mathcal{P}_\eta^{(t)}) - \frac{\langle r_\ell, r_\ell(\mathcal{P}_\eta^{(t)})\rangle}{\|r_\ell\|_2^2} r_\ell \right\|_1 < \frac{\varepsilon_{\mathit{stop}}}{2d} \tag{5.8}$$

is bounded by

$$t \le 8d^2(\sqrt{n}+1)^2 \varepsilon_{\mathit{stop}}^{-2} \log\Big(\eta^{-1}\|\exp(-\mathcal{C}/\eta)\|_1\Big).$$

When the alternative stopping criterion (5.8) *is satisfied, the stopping criterion* (5.5) *is also satisfied.*

A transport plan $\hat{\mathcal{P}} \in B(r_1,\dots,r_d)$ is called ε-approximate solution of the original problem (5.1) if it satisfies

$$\langle C, \hat{\mathcal{P}}\rangle \le \min_{\mathcal{P}\in B(r_1,\dots,r_d)} \langle \mathcal{C}, \mathcal{P}\rangle + \varepsilon.$$

The transport plan $\mathcal{P}_\eta^{(t)}$ obtained from Algorithm 11 using either stopping criterion from Theorem 5.1 is, in general, not in $B(r_1,\dots,r_d)$ because the marginal constraints $r_\ell = r_\ell(\mathcal{P}_\varepsilon^{(t)})$ are satisfied simultaneously only in the limit $t \to \infty$. To fix this issue, rounding [9] can be applied to enforce the marginal constraints on $P_\varepsilon^{(t)}$ for finite t; see Algorithm 12. Note that for a tensor of the form (5.4), the operation in line 5 of Algorithm 12 can be phrased in terms of modifying $\beta_\ell^{(t)}$. Line 6 is a rank-1 update. In [127, Lemma 3.6] and in [217, Theorem 4.4], the following property of the resulting tensor is proven.

Lemma 5.1. *Let* $\mathcal{A} \in \mathbb{R}_{>0}^{n_1\times\cdots\times n_d}$ *and* $r_\ell \in \Delta^{n_\ell}$, $\ell = 1,\dots,d$. *Let* $\mathcal{B}$ *denote the output of Algorithm 12 applied to* $\mathcal{A}$ *and* $r_1,\dots,r_d$. *Then* $\mathcal{B} \in B(r_1,\dots,r_d)$ *and*

$$\|\mathcal{A}-\mathcal{B}\|_1 \le 2\sum_{\ell=1}^{d} \|r_\ell - r_\ell(\mathcal{A})\|_1.$$

Algorithm 12 Rounding

```
Input: tensor $\mathcal{A} \in \mathbb{R}_{>0}^{n_1 \times \cdots \times n_d}$, vectors $r_\ell \in \Delta^{n_\ell}$ for $\ell = 1, \ldots, d$
Output: tensor $\mathcal{B} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$
for $\ell = 1, \ldots, d$
    $v = \min(r_\ell(\mathcal{A})^{-1} * r_\ell, \mathbf{1}_{n_\ell})$, where the min is taken elementwise
    $\mathcal{A} = \mathcal{A} \times_\ell \operatorname{diag}(v)$
$\mathcal{B} = \mathcal{A} + \|r_1 - r_1(\mathcal{A})\|_1^{-(d-1)} \bigcirc_{\ell=1}^{d} (r_\ell - r_\ell(\mathcal{A}))$, where $\bigcirc$ denotes the outer product
```

By combining Algorithm 12 with Algorithm 11, using the index selection (5.7) and the stopping criterion (5.5), it follows [127, Corollary 3.8] that an ε-approximate solution can be computed in

$$\mathcal{O}(\varepsilon^{-3} d^4 n^{d+1} \log(n) (\max(\mathcal{C}) - \min(\mathcal{C}))^3)$$

operations, where $n_1 = \cdots = n_d = n$. In practice, the computation can be accelerated by using slightly modified marginals and tolerances in the Sinkhorn algorithm [217], which ensure that the upper bound (5.6) for t is not dominated by small entries in r_ℓ.

5.2 Impact of approximating the Gibbs kernel

To accelerate the computation of marginals in the multi-marginal Sinkhorn algorithm, we will replace the Gibbs kernel $\mathcal{K}$ by an approximation $\tilde{\mathcal{K}}$; thus replacing Algorithm 11 by Algorithm 13. In this section, we will analyze the impact of this approximation on the transport cost of the computed transport plan. For $d = 2$, such an analysis can be found in [10, Theorem 5]. In the following theorem, we generalize this result to the multi-marginal setting. Our proof closely follows the ideas in [10]. In contrast to the asymptotic result in [7, Theorem 7.4], we provide explicit bounds.

Algorithm 13 Multi-marginal Sinkhorn algorithm for a Gibbs kernel approximation

```
Input: approximation $\tilde{\mathcal{K}} \in \mathbb{R}_{>0}^{n_1 \times n_2 \cdots \times n_d}$ of Gibbs kernel $\mathcal{K} = \exp(-\mathcal{C}/\eta)$, marginals $r_1, \ldots, r_d$
Output: transport plan $\tilde{\mathcal{P}}^{(t)}$
$\beta_\ell^{(0)} = \mathbf{0} \in \mathbb{R}^{n_\ell}$ for $\ell = 1, \ldots, d$
for $t = 0, 1, \ldots$ until stopping criterion (5.5) is satisfied
    $\tilde{\mathcal{P}}^{(t)} = \tilde{\mathcal{K}} \times_1 \operatorname{diag}(\exp(\beta_1^{(t)})) \cdots \times_d \operatorname{diag}(\exp(\beta_d^{(t)}))$
    Let $\ell_{\mathsf{next}}$ denote the index of the scaling parameter that should be updated next.
    $\beta_\ell^{(t+1)} = \begin{cases} \log(r_\ell) - \log(r_\ell(\tilde{\mathcal{P}}^{(t)})) + \beta_\ell^{(t)} & \ell = \ell_{\mathsf{next}} \\ \beta_\ell^{(t)} & \ell \neq \ell_{\mathsf{next}} \end{cases}$
```

Theorem 5.2. *Let* $\mathcal{K} = \exp(-\mathcal{C}/\eta)$ *and assume that* $\tilde{\mathcal{K}} \in \mathbb{R}_{>0}^{n_1 \times n_2 \cdots \times n_d}$ *with* $n_\ell \geq 2$ *and*

$d \geq 2$, satisfies $\|\log(\mathcal{K}) - \log(\tilde{\mathcal{K}})\|_\infty \leq \varepsilon_{\mathsf{log}} \leq 1$. Let $\tilde{\mathcal{P}}$ denote the transport plan returned by Algorithm 13 with stopping criterion $\sum_{\ell=1}^{d} \|r_\ell(\tilde{\mathcal{P}}) - r_\ell\|_1 \leq \varepsilon_{\mathsf{stop}}$. Then

$$|V_{\mathcal{C}}^{\eta}(\mathcal{P}_\eta^*) - V_{\mathcal{C}}^{\eta}(\tilde{\mathcal{P}})| \leq \varepsilon_{V_{\mathcal{C}}^{\eta}},$$

where $\mathcal{P}_\eta^* = \arg\min_{\mathcal{P} \in B(r_1,\dots,r_d)} V_{\mathcal{C}}^{\eta}(\mathcal{P})$ and

$$\begin{aligned}\varepsilon_{V_{\mathcal{C}}^{\eta}} =& \eta\Big(\varepsilon_{\mathsf{log}}\Big(2 + \log\Big(\frac{2}{\varepsilon_{\mathsf{log}}}\Big)\Big) + \frac{\varepsilon_{\mathsf{log}}}{2}\log\Big(\Big(\prod_{\ell=1}^{d} n_\ell\Big) - 1\Big) \\ &+ 2\varepsilon_{\mathsf{stop}}\log\Big(\frac{1}{\varepsilon_{\mathsf{stop}}}\Big(\Big(\prod_{\ell=1}^{d} n_\ell\Big) - 1\Big)\Big)\Big) + (\varepsilon_{\mathsf{log}} + 2\varepsilon_{\mathsf{stop}})\|\mathcal{C}\|_\infty. \end{aligned} \tag{5.9}$$

Proof. We denote by Π^S the operator mapping a given tensor $\mathcal{T} \in \mathbb{R}_{>0}^{n_1 \times \cdots \times n_d}$ to its unique [126] tensor scaling $\mathcal{U} \in B(r_1, \dots, r_d)$ of the form $\mathcal{U} = \mathcal{T} \times_1 \operatorname{diag}(\gamma_1) \times_2 \cdots \times_d \operatorname{diag}(\gamma_d)$ for some vectors $\gamma_\ell \in \mathbb{R}_{>0}^{n_\ell}$ for $\ell = 1, \dots, d$. Observe that $\mathcal{P}_\eta^* = \Pi^S(\mathcal{K})$. Using the triangle inequality, we decompose the error into

$$\begin{aligned} |V_{\mathcal{C}}^{\eta}(\mathcal{P}_\eta^*) - V_{\mathcal{C}}^{\eta}(\tilde{\mathcal{P}})| \leq\ & |V_{\mathcal{C}}^{\eta}(\Pi^S(\mathcal{K})) - V_{\mathcal{C}}^{\eta}(\Pi^S(\tilde{\mathcal{K}}))| && (5.10)\\ &+ |V_{\mathcal{C}}^{\eta}(\Pi^S(\tilde{\mathcal{K}})) - V_{\tilde{\mathcal{C}}}^{\eta}(\Pi^S(\tilde{\mathcal{K}}))| && (5.11)\\ &+ |V_{\tilde{\mathcal{C}}}^{\eta}(\Pi^S(\tilde{\mathcal{K}})) - V_{\tilde{\mathcal{C}}}^{\eta}(\tilde{\mathcal{P}})| && (5.12)\\ &+ |V_{\tilde{\mathcal{C}}}^{\eta}(\tilde{\mathcal{P}}) - V_{\mathcal{C}}^{\eta}(\tilde{\mathcal{P}})| && (5.13)\end{aligned}$$

where $\tilde{\mathcal{C}} = -\eta\log(\tilde{\mathcal{K}})$. We derive bounds for each of these terms; their combination yields inequality (5.9).

Bound for (5.10)**:** By definition of $V_{\mathcal{C}}^{\eta}$, we have

$$\begin{aligned}&|V_{\mathcal{C}}^{\eta}(\Pi^S(\mathcal{K})) - V_{\mathcal{C}}^{\eta}(\Pi^S(\tilde{\mathcal{K}}))| \leq \\ &\quad \|\Pi^S(\mathcal{K}) - \Pi^S(\tilde{\mathcal{K}})\|_1\|\mathcal{C}\|_\infty + \eta|H(\Pi^S(\mathcal{K})) - H(\Pi^S(\tilde{\mathcal{K}}))|.\end{aligned}$$

Because of $\underset{\mathcal{P} \in B(r_1,\dots,r_d)}{\arg\min}\ V_{\mathcal{C}}^{\eta}(\mathcal{P}) = \underset{\mathcal{P} \in B(r_1,\dots,r_d)}{\arg\min}\ \langle -\log(\mathcal{K}), \mathcal{P}\rangle - H(\mathcal{P})$, it follows that

$$\begin{aligned}&\|\Pi^S(\mathcal{K}) - \Pi^S(\tilde{\mathcal{K}})\|_1 = \\ &\quad \Big\| \underset{\mathcal{P} \in B(r_1,\dots,r_d)}{\arg\min}\ \langle -\log(\mathcal{K}), \mathcal{P}\rangle - H(\mathcal{P}) - \Big(\underset{\tilde{\mathcal{P}} \in B(r_1,\dots,r_d)}{\arg\min}\ \langle -\log(\tilde{\mathcal{K}}), \tilde{\mathcal{P}}\rangle - H(\tilde{\mathcal{P}})\Big)\Big\|_1.\end{aligned}$$

Applying Lemma I in [10] to the right hand side of this expression yields $\|\Pi^S(\mathcal{K}) - \Pi^S(\tilde{\mathcal{K}})\|_1 \leq \|\log\mathcal{K} - \log\tilde{\mathcal{K}}\|_\infty \leq \varepsilon_{\mathsf{log}}$. From Theorem 6 in [169] and Lemma D in [10] we obtain

$$|H(\Pi^S(\mathcal{K})) - H(\Pi^S(\tilde{\mathcal{K}}))| \leq \varepsilon_{\mathsf{log}} \log\Big(\frac{2}{\varepsilon_{\mathsf{log}}}\Big) + \frac{\varepsilon_{\mathsf{log}}}{2} \log\Big(\Big(\prod_{\ell=1}^{d} n_\ell\Big) - 1\Big).$$

Bound for (5.11) **and** (5.13)**:** Using that $\|\Pi^S(\tilde{\mathcal{K}})\|_1 = \|\tilde{\mathcal{P}}\|_1 = 1$ we obtain

$$|V_{\mathcal{C}}^{\eta}(\Pi^S(\tilde{\mathcal{K}})) - V_{\tilde{\mathcal{C}}}^{\eta}(\Pi^S(\tilde{\mathcal{K}}))| \leq \langle \mathcal{C}, \Pi^S(\tilde{\mathcal{K}})\rangle - \langle \tilde{\mathcal{C}}, \Pi^S(\tilde{\mathcal{K}})\rangle \leq \|\mathcal{C} - \tilde{\mathcal{C}}\|_\infty \leq \eta\varepsilon_{\mathsf{log}},$$
$$|V_{\tilde{\mathcal{C}}}^{\eta}(\tilde{\mathcal{P}}) - V_{\mathcal{C}}^{\eta}(\tilde{\mathcal{P}})| \leq \langle \mathcal{C}, \tilde{\mathcal{P}}\rangle - \langle \tilde{\mathcal{C}}, \tilde{\mathcal{P}}\rangle \leq \|\mathcal{C} - \tilde{\mathcal{C}}\|_\infty \leq \eta\varepsilon_{\mathsf{log}}.$$

Bound for (5.12)**:** Using that the tensor $\tilde{\mathcal{P}}$ is the unique minimizer of $\underset{\mathcal{P}\in B(r_1(\tilde{\mathcal{P}}),\dots,r_d(\tilde{\mathcal{P}}))}{\arg\min} V_{\tilde{\mathcal{C}}}^{\eta}(\mathcal{P})$, Lemma H in [10] yields

$$|V_{\tilde{\mathcal{C}}}^{\eta}(\Pi^S(\tilde{\mathcal{K}})) - V_{\tilde{\mathcal{C}}}^{\eta}(\tilde{\mathcal{P}})| \leq \omega(d_H(B(r_1(\tilde{\mathcal{P}}),\dots,r_d(\tilde{\mathcal{P}})), B(r_1,\dots,r_d))),$$

where $d_H(\cdot,\cdot)$ denotes the Hausdorff distance and

$$\omega(x) = x\|\mathcal{C}\|_\infty + \eta\Big(x \log\Big(\frac{2}{x}\Big(\Big(\prod_{\ell=1}^{d} n_\ell\Big) - 1\Big)\Big)\Big).$$

We can bound $d_H(B(r_1(\tilde{\mathcal{P}}),\dots,r_d(\tilde{\mathcal{P}})), B(r_1,\dots,r_d))$ by $2\varepsilon_{\mathsf{stop}}$, since Algorithm 12 maps any $\mathcal{A} \in B(r_1(\tilde{\mathcal{P}}),\dots,r_d(\tilde{\mathcal{P}}))$ to $\mathcal{B} \in B(r_1,\dots,r_d)$ with $||\mathcal{A} - \mathcal{B}||_1 \leq 2\varepsilon_{\mathsf{stop}}$ as stated in Lemma 5.1. This implies

$$|V_{\tilde{\mathcal{C}}}^{\eta}(\Pi^S(\tilde{\mathcal{K}})) - V_{\tilde{\mathcal{C}}}^{\eta}(\tilde{\mathcal{P}})| \leq 2\varepsilon_{\mathsf{stop}}\|\mathcal{C}\|_\infty + 2\eta\varepsilon_{\mathsf{stop}} \log\Big(\frac{1}{\varepsilon_{\mathsf{stop}}}\Big(\Big(\prod_{\ell=1}^{d} n_\ell\Big) - 1\Big)\Big).$$

□

The following theorem demonstrates how the previous result can be combined with Algorithm 12 to obtain ε-accurate solutions. The proof is inspired by [217, Theorem 4.5] and [127, Theorem 3.7], which state how ε-accurate solutions can be computed using Algorithm 11. Theorem 5.3 takes into account that we compute the transport plan using Algorithm 13 based on a perturbed Gibbs kernel.

Theorem 5.3. *Let $\hat{\mathcal{P}}$ be the tensor obtained by applying Algorithm 12 to $\tilde{\mathcal{P}}$, where $\tilde{\mathcal{P}}$ is obtained from Algorithm 13 with $\tilde{\mathcal{K}}$ fulfilling the assumptions of Theorem 5.2. Let $\mathcal{P}^* = \arg\min_{\mathcal{P}\in B(r_1,\dots,r_d)}\langle \mathcal{C}, \mathcal{P}\rangle$. Then it holds*

$$\langle \mathcal{C}, \hat{\mathcal{P}}\rangle - \langle \mathcal{C}, \mathcal{P}^*\rangle \leq \varepsilon,$$

where $\varepsilon = 2\eta\varepsilon_{log} + 2\eta\sum_{\ell=1}^{d}\log(n_\ell) + 4\|\mathcal{C}\|_\infty\varepsilon_{stop}$.

Proof. From $\mathcal{P}^* \in B(r_1,\dots,r_d)$ and $\mathcal{P}^*_\eta = \arg\min_{\mathcal{P}\in B(r_1,\dots,r_d)}\langle\mathcal{C},\mathcal{P}\rangle - \eta H(\mathcal{P})$ follows $\langle\mathcal{C},\mathcal{P}^*_\eta\rangle - \eta H(\mathcal{P}^*_\eta) \le \langle\mathcal{C},\mathcal{P}^*\rangle - \eta H(\mathcal{P}^*)$. Thus,

$$\langle\mathcal{C},\mathcal{P}^*_\eta\rangle - \langle\mathcal{C},\mathcal{P}^*\rangle \le \eta H(\mathcal{P}^*_\eta) - \eta H(\mathcal{P}^*) \le \eta\sum_{\ell=1}^{d}\log(n_\ell), \tag{5.14}$$

where we use that $0 \le H(\mathcal{X}) \le \sum_{\ell=1}^{d}\log(n_\ell)$ for any tensor $\mathcal{X} \in \mathbb{R}_+^{n_1\cdots n_d}$ with $||\mathcal{X}||_1 = 1$ [81, Theorem 2.6.4].

Note that the marginals of $\tilde{\mathcal{P}}$ are, in general, not equal to $r_1,\dots,r_d$. In order to compare $\tilde{\mathcal{P}}$ and $\mathcal{P}^*_\eta$, we construct $\mathcal{Q} \in B(r_1(\tilde{\mathcal{P}}),\dots,r_d(\tilde{\mathcal{P}}))$ by applying Algorithm 12 to $\mathcal{P}^*_\eta$ with marginals $r_1(\tilde{\mathcal{P}}),\dots,r_d(\tilde{\mathcal{P}})$. Since $\mathcal{P}^*_\eta \in B(r_1,\dots,r_d)$, Lemma 5.1 implies that $||\mathcal{Q} - \mathcal{P}^*_\eta||_1 \le 2\varepsilon_{\mathsf{stop}}$. Hence,

$$\langle\mathcal{C},Q\rangle - \langle\mathcal{C},\mathcal{P}^*_\eta\rangle \le \|\mathcal{C}\|_\infty\|\mathcal{Q} - \mathcal{P}^*_\eta\|_1 \le 2\|\mathcal{C}\|_\infty\varepsilon_{\mathsf{stop}}. \tag{5.15}$$

The tensor $\tilde{\mathcal{P}}$ is the unique scaling of $\tilde{\mathcal{K}}$ with marginals $r_1(\tilde{\mathcal{P}}),\dots,r_d(\tilde{\mathcal{P}})$ [26]. It is thus the unique minimizer [127] of $\arg\min_{\mathcal{P}\in B(r_1(\tilde{\mathcal{P}}),\dots,r_d(\tilde{\mathcal{P}}))}\langle\tilde{\mathcal{C}},\mathcal{P}\rangle - \eta H(\mathcal{P})$, where $\tilde{\mathcal{C}} = -\eta\log(\tilde{\mathcal{K}})$. Following the same argument as in (5.14), we obtain

$$\langle\tilde{\mathcal{C}},\tilde{\mathcal{P}}\rangle - \langle\tilde{\mathcal{C}},\mathcal{Q}\rangle \le \eta H(\tilde{\mathcal{P}}) - \eta H(\mathcal{Q}) \le \eta\sum_{\ell=1}^{d}\log(n_\ell). \tag{5.16}$$

We further obtain

$$\langle\tilde{\mathcal{C}},\mathcal{Q}\rangle - \langle\mathcal{C},\mathcal{Q}\rangle \le \|\tilde{\mathcal{C}} - \mathcal{C}\|_\infty\|\mathcal{Q}\|_1 = \|\tilde{\mathcal{C}} - \mathcal{C}\|_\infty \le \eta\varepsilon_{\mathsf{log}}, \tag{5.17}$$

$$\langle\mathcal{C},\tilde{\mathcal{P}}\rangle - \langle\tilde{\mathcal{C}},\tilde{\mathcal{P}}\rangle \le \|\tilde{\mathcal{C}} - \mathcal{C}\|_\infty\|\tilde{\mathcal{P}}\|_1 = \|\tilde{\mathcal{C}} - \mathcal{C}\|_\infty \le \eta\varepsilon_{\mathsf{log}}, \tag{5.18}$$

where we use that $\|\tilde{\mathcal{P}}\|_1 = \|\mathcal{Q}\|_1 = 1$. Additionally, Lemma 5.1 yields

$$\langle\mathcal{C},\hat{\mathcal{P}}\rangle - \langle\mathcal{C},\tilde{\mathcal{P}}\rangle \le \|\mathcal{C}\|_\infty\|\hat{\mathcal{P}} - \tilde{\mathcal{P}}\|_1 \le 2\|\mathcal{C}\|_\infty\varepsilon_{\mathsf{stop}}. \tag{5.19}$$

By adding the inequalities (5.14)–(5.19) we obtain

$$\langle\mathcal{C},\hat{\mathcal{P}}\rangle - \langle\mathcal{C},\mathcal{P}^*\rangle \le 2\eta\varepsilon_{\mathsf{log}} + 2\eta\sum_{\ell=1}^{d}\log(n_\ell) + 4\|\mathcal{C}\|_\infty\varepsilon_{\mathsf{stop}}.$$

□

Remark 5.1. *Note that the* $\eta\varepsilon_{log}$ *only converges to zero in the limit* $\eta \to 0$ *when the approximation* $\tilde{\mathcal{K}}$ *is equal to the true Gibbs kernel* $\mathcal{K}$.

Remark 5.2. *Note that for any* $\varepsilon > 0$*, we can find suitable* $\eta, \varepsilon_{\mathsf{log}}, \varepsilon_{\mathsf{stop}}$ *such that combining Algorithm 13 and Algorithm 12 yields an* ε*-accurate solution for the multi-marginal optimal transport problem* (5.1)*. Analogously, we can find* $\eta, \varepsilon_{\mathsf{log}}, \varepsilon_{\mathsf{stop}}$ *such that* (5.9) *is satisfied for any given* $\varepsilon_{V_{\mathcal{C}}^{\eta}} > 0$*. In particular, we can first select* $\varepsilon_{\mathsf{log}}$ *and* $\varepsilon_{\mathsf{stop}}$ *based on* $\|\mathcal{C}\|_{\infty}$*. Afterwards, we can set* η *sufficiently small.*

Remark 5.3. *It might seem counter-intuitive that* $\varepsilon_{\mathsf{log}}$ *and* $\varepsilon_{\mathsf{stop}}$ *needs to be chosen inversely proportional to* $\|\mathcal{C}\|_{\infty}$ *in the previous remark. This is caused by the chosen objective function in* (5.2)*. Let* $c = \|C\|_{\infty}^{-1}$*. Note that optimal solution of the regularized* (5.2) *and the set of optimal solutions of the original optimal transport problem* (5.1) *do not change when we replace both* $\mathcal{C}$ *by* $c\mathcal{C}$ *and* η *by* $c\eta$*. This normalization changes the optimal value of the objective functions in* (5.2) *and* (5.1) *by* c*, but it does not change the Gibbs kernel* $\mathcal{K}$*. Thus, we can transform the error bounds in Theorem 5.2 and 5.3 into bounds for this normalized problem, by multiplying* $\varepsilon_{V_{\mathcal{C}}^{\eta}}$ *and* ε *by* c*. To obtain a certain* $c\varepsilon_{V_{\mathcal{C}}^{\eta}}$ *respectively* $c\varepsilon$*, we can select* $\varepsilon_{\mathsf{log}}$ *and* $\varepsilon_{\mathsf{stop}}$ *independently from* $\|\mathcal{C}\|_{\infty}$ *before determining a suitable* η*.*

5.3 Tensor networks and graphical models

In applications, the cost tensor $\mathcal{C} \in \mathbb{R}_{+}^{n_1 \times \cdots \times n_d}$ usually carries additional structure. A broad class of structures leads to transport plans defined via graphical models [150]. In this case, the entries of $\mathcal{C}$ take the form

$$\mathcal{C}_I = \sum_{\alpha \in F} \mathcal{C}_{I_\alpha}^{\alpha} \quad \text{for every } I = (i_1, \dots, i_d), \quad i_\ell = 1, \dots, n_\ell,\ \ell = 1, \dots, d, \tag{5.20}$$

where the summation index tuples $\alpha = (\alpha_1, \dots, \alpha_M)$ are contained in a fixed subset F of

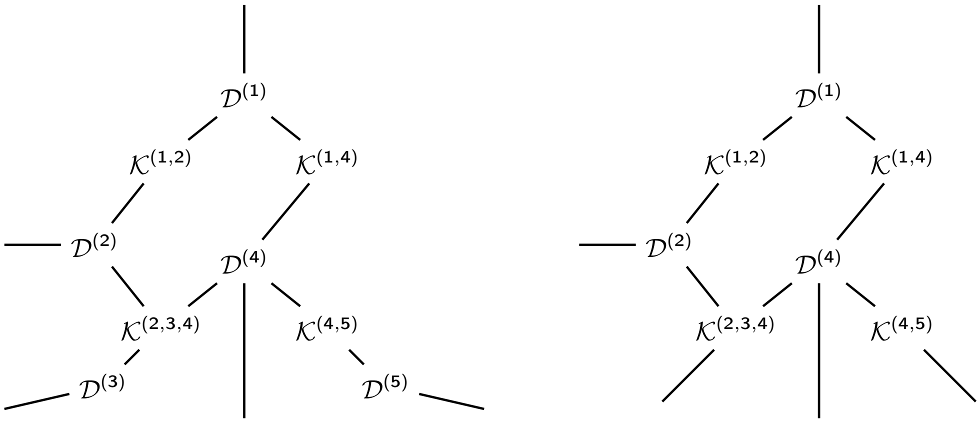
$$\bigcup_{M=1}^{d} \{(\alpha_1, \dots, \alpha_M) \in \mathbb{N}^M | 1 \leq \alpha_1 < \cdots < \alpha_M \leq d\},$$

$I_\alpha := (i_{\alpha_1}, \dots, i_{\alpha_M})$ and $\mathcal{C}^{\alpha} \in \mathbb{R}_{+}^{n_{\alpha_1} \times \cdots \times n_{\alpha_M}}$. The corresponding Gibbs kernel is given by

$$\mathcal{K}_I = \prod_{\alpha \in F} \mathcal{K}_{I_\alpha}^{\alpha} \quad \text{for every } I = (i_1, \dots, i_d), \quad i_\ell = 1, \dots, n_\ell,\ \ell = 1, \dots, d, \tag{5.21}$$

where $\mathcal{K}^{\alpha} = \exp(-\mathcal{C}^{\alpha}/\eta)$. This Gibbs kernel can be represented in terms of a tensor network [247]. In general, a tensor network represents a high-order tensor that is constructed by contracting several low-order tensors. By contraction we refer to the sum over a joint index in two low-order tensors. A graph is used to describe precisely how the low-order tensors should to be contracted. Its vertices correspond to the low-order tensors. Each edge corresponds to a contraction of the two low-order tensors corresponding to the vertices connected by the edge.

In the following, we describe how to construct the particular network for $\mathcal{K}$. First, we add each tensor $\mathcal{K}^{\alpha}$ as a vertex. For every $\ell = 1, \dots, d$, we add an additional tensor $\mathcal{D}^{(\ell)}$ as vertex. For every $\ell = 1, \dots, d$ and $\alpha \in F$, we add edges from $\mathcal{K}^{\alpha}$ to $\mathcal{D}^{(\ell)}$ if ℓ is contained in α. We then add one open edge to each $\mathcal{D}^{(\ell)}$ that corresponds to the index in the ℓth mode of $\mathcal{K}$. The order d_ℓ of the tensors $\mathcal{D}^{(\ell)} \in \mathbb{R}^{n_\ell \times \cdots \times n_\ell}$ is equal to the number of connected edges. Their entries are given by $\mathcal{D}^{(\ell)}_{i_1,\dots,i_{d_\ell}} = \delta_{i_1,i_2}\delta_{i_2,i_3} \cdots \delta_{i_{d_\ell - 1},i_{d_\ell}}$, where δ denotes the Kronecker delta. Each edge in the tensor network corresponds to the sum over the corresponding index in the connected vertices. See Figure 5.1 for an example.



(a) Direct construction

(b) After simplification

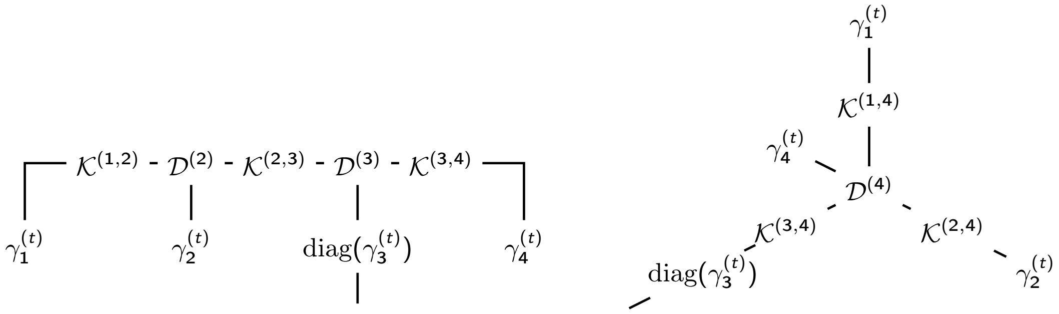
Figure 5.1 – Tensor network representation of $\mathcal{K}$ for a cost tensor of the form (5.20) constructed from $\mathcal{C}^{(1,2)}, \mathcal{C}^{(1,4)}\mathcal{C}^{(2,3,4)}, \mathcal{C}^{(4,5)}$. In (b), we slightly simplified the network by contracting the identity matrices $\mathcal{D}^{(3)}, \mathcal{D}^{(5)}$ with their connected open edges. Summing over all internal edges yields the elementwise representation

$$\mathcal{K}_{i_1,i_2,i_3,i_4,i_5} = \sum_{j_1=1}^{n_1}\sum_{j_2=1}^{n_1}\sum_{j_3=1}^{n_2}\sum_{j_4=1}^{n_2}\sum_{j_5=1}^{n_4}\sum_{j_6=1}^{n_4}\sum_{j_7=1}^{n_4} \mathcal{D}^{(1)}_{i_1,j_1,j_2}\mathcal{D}^{(2)}_{i_2,j_3,j_4}\mathcal{D}^{(4)}_{i_4,j_5,j_6,j_7}\mathcal{K}^{(1,2)}_{j_1,j_3}\mathcal{K}^{(1,4)}_{j_2,j_5}\mathcal{K}^{(2,3,4)}_{j_4,i_3,j_6}\mathcal{K}^{(4,5)}_{j_7,i_5}.$$

Remark 5.4. *We want to emphasize that this tensor network is closely related to the dual tensor network of the graphical model representing the transport plan [273]. In the context of graphical models, the cost tensors $\mathcal{C}$ is given in the form of* (5.20)*. The resulting transport plan $\mathcal{P}_\eta^{(t)}$ defined in* (5.3) *can be represented as tensor network by attaching the matrices diag*$(\exp(\beta_\ell))$ *to the corresponding open modes of the tensor network representation of $\mathcal{K}$. At the same time, $\mathcal{P}_\eta^{(t)}$ represents a discrete probability distribution, which can be represented by a graphical model [150]. This graphical model and the tensor network of $\mathcal{P}_\eta^{(t)}$ are duals of each other [273].*

We can compute $\mathcal{K}$ from the tensor network by contracting each of the internal edges sequentially. The contraction of one internal edge corresponds to the merging the connected vertices by evaluating of the sum over the index corresponding to the edge [247]. When the tensor network contains circles, multi-edges will occur, which can be contracted by summing over all the corresponding indices simultaneously. The order of contracting the internal edges determines the degree of the occurring intermediate vertices and the computational complexity. In our constructed tensor network, we can exploit the special structure of $\mathcal{D}^{(\ell)}$ to contract all connected edges simultaneously. The book [272] discusses several heuristics to optimize the order of contractions. Note that the optimal contraction sequence might still incur a large computational cost.

In Algorithm 11, we need to evaluate the marginals of $\mathcal{P}_\eta^{(t)}$ in each iteration. Let $\gamma_\ell^{(t)} = \exp(\beta_\ell^{(t)})$ for $\ell = 1, \dots, d$. Given a tensor network representation of $\mathcal{K}$, the mode-ℓ marginals can be written as a contraction of the network after connecting the matrix $\mathrm{diag}(\gamma_\ell^{(t)})$ to the open edge corresponding to mode ℓ, and the vectors $\gamma_{\tilde{\ell}}^{(t)}$ for $\tilde{\ell} \neq \ell$ to their respective open edges. By contracting all inner edges of the resulting network, we obtain $r_\ell(\mathcal{P}_\eta^{(t)})$. We refer to Figure 5.2 for examples. The depicted networks will again be used in the numerical experiments in Section 5.5.



(a) (b)

Figure 5.2 – Examples for the tensor network diagram representation of $r_3(\mathcal{P}_\eta^{(t)})$ based on different tensor network structures for the Gibbs kernel (5.21).

We give a brief example on how to efficiently contract the network depicted in Figure 5.2(a). For the complexity analysis we assume $n_1 = n_2 = n_3 = n_4 = n$. We first compute the vectors $v_j^{(1)} = \sum_{i=1}^n (\gamma_1^{(t)})_i \mathcal{K}_{i,j}^{(1,2)}$ and $v_j^{(4)} = \sum_{i=1}^n \mathcal{K}_{j,i}^{(3,4)} (\gamma_4^{(t)})_i$ for $j = 1, \dots, n$ in $n(2n-1)$ operations each, where operations refers to the required number of additions and multiplications. This corresponds to contracting the edge between $\gamma_1^{(t)}$ and $\mathcal{K}^{(1,2)}$ as well as the edge between $\mathcal{K}^{(3,4)}$ and $\gamma_4^{(t)}$. In the next step, we contract the edges from $\mathcal{D}^{(2)}$ to $v^{(1)}, \gamma_2^{(t)}$ and $\mathcal{K}^{(2,3)}$ simultaneously. This corresponds to computing the vector $v_j^{(2)} = \sum_{i=1}^n (v^{(1)} * \gamma_2^{(t)})_i \mathcal{K}_{i,j}^{(2,3)}$ for $j = 1, \dots, n$ in $n + n(2n-1) = 2n^2$ operations, where we first compute the elementwise product before evaluating the matrix vector product. Finally, we contract the remaining edges around $\mathcal{D}^{(3)}$ to compute $r_3(\mathcal{P}_\eta^{(t)}) = v^{(2)} * \gamma_3^{(t)} * v^{(4)}$ in $2n$ operations. In total, we need $6n^2$ operations to compute the marginal $r_3(\mathcal{P}_\eta^{(t)})$. Note that this contraction strategy corresponds to the following reordering of the sums in the definition of the marginal

$$
\begin{aligned}
r_3(P_\eta^{(t)})_{i_3} &= \sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\sum_{i_4=1}^{n} \mathcal{K}^{(1,2)}_{i_1,i_2}\mathcal{K}^{(2,3)}_{i_2,i_3}\mathcal{K}^{(3,4)}_{i_3,i_4}(\gamma_1^{(t)})_{i_1}(\gamma_2^{(t)})_{i_2}(\gamma_3^{(t)})_{i_3}(\gamma_4^{(t)})_{i_4} \\
&= (\gamma_3^{(t)})_{i_3}\left(\sum_{i_2=1}^{n}\mathcal{K}^{(2,3)}_{i_2,i_3}\left((\gamma_2^{(t)})_{i_2}\left(\sum_{i_1=1}^{n}\mathcal{K}^{(1,2)}_{i_1,i_2}(\gamma_1^{(t)})_{i_1}\right)\right)\right)\left(\sum_{i_4=1}^{n}\mathcal{K}^{(3,4)}_{i_3,i_4}(\gamma_4^{(t)})_{i_4}\right).
\end{aligned}
$$

We can reuse the intermediate terms $v^{(1)}, v^{(2)}, v^{(4)}$ in the computation of the other marginals. This allows us to compute all four marginals in $12n^2 + 4n$ operations, whereas computing the marginals based on the full tensor requires $\mathcal{O}(n^4)$ operations.

Remark 5.5. *Instead of the tensor network based on an ordinary graph with special tensors* $\mathcal{D}^{(\ell)}$*, we could consider a tensor network based on a hypergraph as in [273]. The structure of* $\mathcal{D}^{(\ell)}$ *can be modeled by a single hyperedge containing all vertices connected to* $\mathcal{D}^{(\ell)}$*. The contraction of these hypergraph based networks is studied in [174].*

Remark 5.6. *Note that the structure of* $\mathcal{K}$ *can also be exploited to compute marginals when applying Algorithm 12 to* $\mathcal{P}_\eta^{(t)}$*. Storing the rank-*1 *update in line 6 separately in terms of its factors offers the potential to avoid storing the transport plan explicitly as a full tensor.*

5.4 Low-rank approximations in tensor networks

Assuming that the Gibbs kernel is represented as in Equation (5.21), we obtain a tensor network containing the coefficient tensors $\mathcal{K}^\alpha$. The following lemma bounds the impact on the Gibbs kernel when replacing each $\mathcal{K}^\alpha$ by an approximation $\tilde{\mathcal{K}}^\alpha$.

Lemma 5.2. *Let* $\mathcal{K}$ *be defined based on tensors* $\mathcal{K}^\alpha \in \mathbb{R}_{>0}^{n_{\alpha_1}\times\cdots\times n_{\alpha_M}}$ *for* $\alpha \in F$ *as in* (5.21)*. Let* $\tilde{\mathcal{K}}^\alpha \in \mathbb{R}_{>0}^{n_{\alpha_1}\times\cdots\times n_{\alpha_M}}$ *for* $\alpha \in F$*. We define* $\tilde{\mathcal{K}} \in \mathbb{R}_{>0}^{n_1\times\cdots\times n_d}$ *elementwise as*

$$
\tilde{\mathcal{K}}_I = \prod_{\alpha\in F}\tilde{\mathcal{K}}^\alpha_{I_\alpha} \qquad \text{for every } I = (i_1,\dots,i_d), \quad i_\ell = 1,\dots,n_\ell,\ \ell = 1,\dots,d. \tag{5.22}
$$

Then

$$
\|\log(\mathcal{K}) - \log(\tilde{\mathcal{K}})\|_\infty \le \sum_{\alpha\in F}\|\log(\mathcal{K}^\alpha) - \log(\tilde{\mathcal{K}}^\alpha)\|_\infty.
$$

Proof.

$$\begin{aligned}\|\log(\mathcal{K}) - \log(\tilde{\mathcal{K}})\|_\infty &= \max_{i_1,\dots,i_d} \Big|\sum_{\alpha\in F} \log(\mathcal{K}^\alpha_{I_\alpha}) - \sum_{\alpha\in F} \log(\tilde{\mathcal{K}}^\alpha_{I_\alpha})\Big| \\ &\le \max_{i_1,\dots,i_d} \sum_{\alpha\in F} |\log(\mathcal{K}^\alpha_{I_\alpha}) - \log(\tilde{\mathcal{K}}^\alpha_{I_\alpha})| \\ &\le \sum_{\alpha\in F} \|\log(\mathcal{K}^\alpha) - \log(\tilde{\mathcal{K}}^\alpha)\|_\infty.\end{aligned}$$

□

Combining Lemma 5.2 and Theorem 5.2 implies that Algorithm 13 with $\tilde{\mathcal{K}}$ defined as in (5.22) yields an accurate approximation of the optimal transport plan when $\|\log(\mathcal{K}^\alpha) - \log(\tilde{\mathcal{K}}^\alpha)\|_\infty$ is sufficiently small for all $\alpha \in F$. This allows one to replace each $\mathcal{K}^\alpha$ by a low-rank approximation $\tilde{\mathcal{K}}^\alpha$, which in turn accelerates the computation of tensor network contractions. For the example at the end of Section 5.3, the number of operations reduces from $\mathcal{O}(n^2)$ to $\mathcal{O}(nr)$ when every $\mathcal{K}^\alpha$ is approximated by a rank-r matrix of the form $\mathcal{K}^\alpha = U^\alpha (V^\alpha)^T$ with $U^\alpha, V^\alpha \in \mathbb{R}^{n\times r}$ as depicted in Figure 5.3(a).

5.5 Numerical experiments

All numerical experiments[*] in this section were performed in MATLAB R2018b on a Lenovo Thinkpad T480s with Intel Core i7-8650U CPU and 15.4 GiB RAM. In Algorithms 11 and 13 we select the next index to be updated using index selection (5.7) and we stop using stopping criterion (5.8) with $\varepsilon_{\mathsf{stop}} = 10^{-4}$.

5.5.1 Proof of concept

In the following, we study the impact of approximating the Gibbs kernel on the transport cost. We define a multi-marginal optimal transport problem, whose cost tensor is of the form studied in [110, Section 5.2]. Let $n = 420$. For $\ell = 1, \dots, 4$, we generate random point sets $X^{(\ell)} = \{x_1^{(\ell)}, \dots, x_n^{(\ell)}\}$ by sampling the points $x_i^{(\ell)} \in \mathbb{R}^2$ independently randomly from the uniform distribution on $[0,1]^2$ for $i = 1, \dots, n$. We define $n \times n$ matrices $C^{(1,2)}, C^{(2,3)}, C^{(3,4)}$ with entries

$$\mathcal{C}^{(1,2)}_{i,j} = \|x_i^{(1)} - x_j^{(2)}\|_2^2,\ \mathcal{C}^{(2,3)}_{j,k} = \|x_j^{(2)} - x_k^{(3)}\|_2^2,\ \mathcal{C}^{(3,4)}_{k,l} = \|x_k^{(3)} - x_l^{(4)}\|_2^2 \quad \text{for } 1 \le i,j,k,l \le n.$$

We construct the cost tensor

$$\mathcal{C}_{i,j,k,l} = \mathcal{C}^{(1,2)}_{i,j} + \mathcal{C}^{(2,3)}_{j,k} + \mathcal{C}^{(3,4)}_{k,l} \quad \text{for } 1 \le i,j,k,l \le n.$$

---

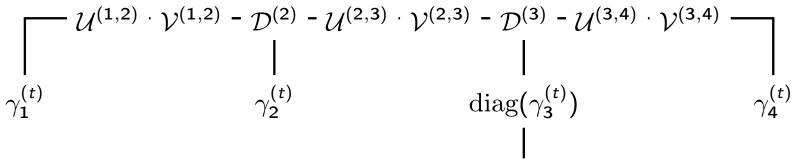
[*] The MATLAB code to reproduce these results is available from https://github.com/cstroessner/Optimal-Transport.git.

Let $\mathcal{K}^\alpha = \exp(-\mathcal{C}^\alpha)$ for $\alpha \in \{(1,2),(2,3),(3,4)\}$. The Gibbs kernel $\mathcal{K} = \exp(-\mathcal{C})$ is represented by the tensor network shown in Figure 5.2(a).

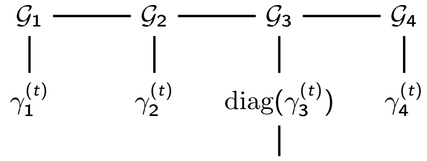
Let $r \le n$. We compare two different approximations of $\mathcal{K}$. For the first one, we replace the matrices $\mathcal{K}^\alpha$ by their rank-r best approximations $\tilde{\mathcal{K}}^\alpha$ using truncated singular value decompositions and define

$$(\mathcal{K}_{\mathsf{SVDs}})_{i,j,k,l} = \tilde{\mathcal{K}}^{(1,2)}_{i,j} \cdot \tilde{\mathcal{K}}^{(2,3)}_{j,k} \cdot \tilde{\mathcal{K}}^{(3,4)}_{k,l} \quad \text{for } 1 \le i,j,k,l \le n.$$

For the second approximation $\mathcal{K}_{\mathsf{TT}}$, we compute a TT approximation 2.6 of $\mathcal{K}$ with TT rank (r,r,r) using the TT-DMRG-cross algorithm [281] ignoring the underlying graph structure. The tensor network representation of $\mathcal{K}_{\mathsf{SVDs}}$ and $\mathcal{K}_{\mathsf{TT}}$ is depicted in Figure 5.3. All four marginals can be computed in $\mathcal{O}(nr)$ operations for $\mathcal{K}_{\mathsf{SVDs}}$ and in $\mathcal{O}(nr^2)$ operations for $\mathcal{K}_{\mathsf{TT}}$ by contracting the tensor networks. In contrast, exploiting the graph structure in $\mathcal{K}$ without low-rank approximations requires $\mathcal{O}(n^2)$ operations.


$\mathcal{U}^{(1,2)} \cdot \mathcal{V}^{(1,2)}$
$\mathcal{D}^{(2)}$
$\mathcal{U}^{(2,3)} \cdot \mathcal{V}^{(2,3)}$
$\mathcal{D}^{(3)}$
$\mathcal{U}^{(3,4)} \cdot \mathcal{V}^{(3,4)}$
$\gamma_1^{(t)}$
$\gamma_2^{(t)}$
$\mathrm{diag}(\gamma_3^{(t)})$
$\gamma_4^{(t)}$

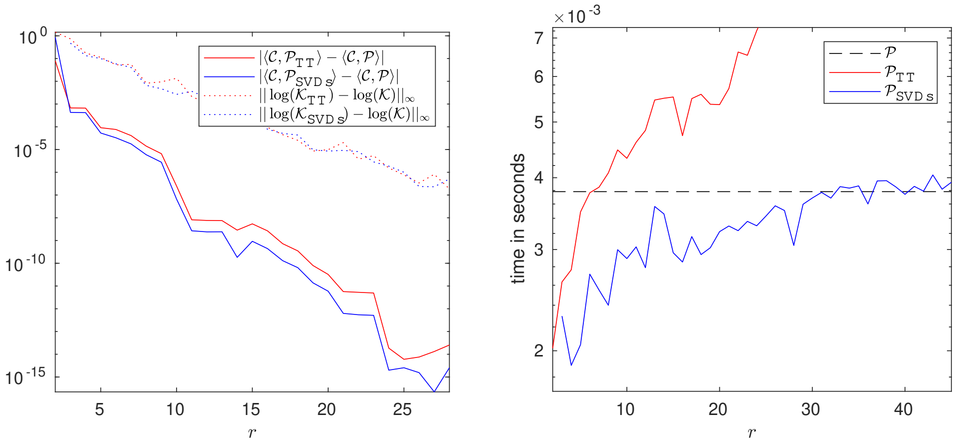
(a) Based on $\mathcal{K}_{\mathsf{SVDs}}$



(b) Based on $\mathcal{K}_{\mathsf{TT}}$

Figure 5.3 – Tensor networks for the computation of $r_3(\mathcal{P}_\eta^{(t)})$ for the example in Section 5.5.1. We express the truncated SVDs in $\mathcal{K}_{\mathsf{SVDs}}$ as $\tilde{\mathcal{K}}^\alpha = U^\alpha (V^\alpha)^T$ with $U^\alpha, V^\alpha \in R^{n\times r}$. The TT cores in $\mathcal{K}_{\mathsf{TT}}$ are denoted by $\mathcal{G}_1 \in \mathbb{R}^{n\times r}$, $\mathcal{G}_2, \mathcal{G}_3 \in \mathbb{R}^{r\times n\times r}$ and $\mathcal{G}_4 \in \mathbb{R}^{r\times n}$.

Based on the tensors $\mathcal{K}, \mathcal{K}_{\mathsf{SVDs}}, \mathcal{K}_{\mathsf{TT}}$, we compute transport plans $\mathcal{P}, \mathcal{P}_{\mathsf{SVDs}}, \mathcal{P}_{\mathsf{TT}}$ by first applying Algorithm 13 with marginals $r_\ell = \frac{1}{n} \cdot \mathbf{1}_n$ and then rounding the resulting tensor using Algorithm 12. In Figure 5.4, we compare the different transport plans. Note that we can efficiently evaluate the transport cost $\langle \mathcal{C}, \mathcal{P} \rangle$ using tensor network contractions of $\mathcal{C}^\alpha$ and $\mathcal{P}$ without evaluating the full tensors. We observe that the difference in transport cost of $\mathcal{P}_{\mathsf{SVDs}}, \mathcal{P}_{\mathsf{TT}}$ and $\mathcal{P}$ is much smaller than the norm of the difference of the logarithms of $\mathcal{K}_{\mathsf{SVDs}}, \mathcal{K}_{\mathsf{TT}}$ and $\mathcal{K}$. The approximation $\mathcal{P}_{\mathsf{SVDs}}$ that exploits the graph structure leads to slightly better approximations compared to $\mathcal{P}_{\mathsf{TT}}$. We want to emphasize that computing $\mathcal{P}_{\mathsf{SVDs}}$ with $r = 25$ is faster than using the graph structure of $\mathcal{K}$ directly and only leads to a difference in transport cost in the order of machine precision. Computing $\mathcal{P}_{\mathsf{TT}}$ is faster than computing $\mathcal{P}$ for very small ranks. The different scaling in the number of operations required to compute marginals leads to larger computation times for $\mathcal{P}_{\mathsf{TT}}$ compared to $\mathcal{P}_{\mathsf{SVDs}}$ for increasing values of r. We want to emphasize that storing a tensor in $\mathbb{R}^{n\times n\times n\times n}$ explicitly would require more than 240GB of memory. Thus, it would not be feasible to solve this problem without exploiting either the underlying structure of $\mathcal{C}$ or the structure of the TT approximation.



(a) Error analysis

(b) Computation time

Figure 5.4 – Difference in transport cost for the example in Section 5.5.1 for various ranks r. Left: We depict the difference in transport cost for $\mathcal{P}_{\mathsf{SVDs}}, \mathcal{P}_{\mathsf{TT}}$ and $\mathcal{P}$ and an estimation of the norm of the difference of the logarithms of $\mathcal{K}_{\mathsf{SVDs}}, \mathcal{K}_{\mathsf{TT}}$ and $\mathcal{K}$ based on 1 000 sample points. Right: Measured computation times for applying Algorithm 13 and 12 to compute the transport plans exploiting the structures of $\mathcal{K}, \mathcal{K}_{\mathsf{SVDs}}, \mathcal{K}_{\mathsf{TT}}$. Note that this time does not include the computation of $\mathcal{K}^{\alpha}, \mathcal{K}_{\mathsf{SVDs}}, \mathcal{K}_{\mathsf{TT}}$.

Remark 5.7. *Figure 5.4 shows that larger ranks lead to smaller approximation errors, but at the same time larger ranks increase the computation time. This needs to be balanced in practice. In particular, the rank needs to be sufficiently large such that the approximation* $\tilde{\mathcal{K}}$ *of the Gibbs kernel is strictly positive, which implies that* $\|\log(\mathcal{K})-\log(\tilde{\mathcal{K}})\|_\infty$ *is bounded. This can be achieved by choosing an approximation such that* $\|\mathcal{K}-\tilde{\mathcal{K}}\|_\infty$ *is smaller than the smallest entry of* $\mathcal{K}$.

Remark 5.8. *The difference of the entropic cost for the tensors* $\mathcal{P}_{\mathsf{SVDs}}, \mathcal{P}_{\mathsf{TT}}$ *and the optimal transport plan* $\mathcal{P}^*_\eta$ *is bounded by Theorem 5.2. We can assume that* $\mathcal{P}$ *is a good approximation of* $\mathcal{P}^*_\eta$*. This would allow us to study the sharpness of the bound numerically. However, the evaluation of the entropic cost requires the explicit computation of the full tensors, which is not feasible for* $n = 420$ *without a Monte Carlo approximation. Instead, we repeat the experiment in Section 5.5.1 with a smaller* n*. The results are depicted in Figure 5.5. We find that the theoretical bound is much larger than the error observed in practice.*

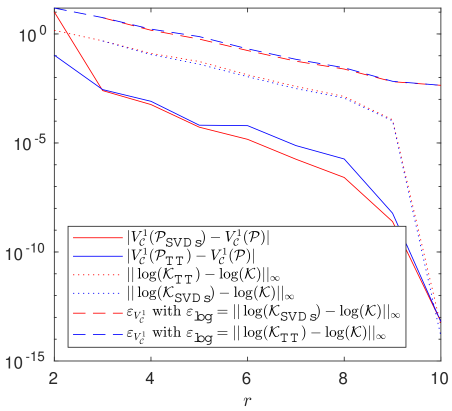


Figure 5.5 – We repeat the experiment described in Section 5.5.1 with $n = 10$ for various ranks r. We depict the difference in the entropic transport cost (5.2) of $\mathcal{P}_{\mathsf{SVDs}}, \mathcal{P}_{\mathsf{TT}}$ and $\mathcal{P}$. Further, we depict the norm of the difference of the elementwise logarithm of $\mathcal{K}_{\mathsf{SVDs}}, \mathcal{K}_{\mathsf{TT}}$ and $\mathcal{K}$. Based on these values we compute the value of $\varepsilon_{V_\mathcal{C}^1}$ as defined in (5.9).

5.5.2 Application: Color transfer from color barycenters

In the following, we apply our algorithms for color transfer as in [150]. We consider $d = 4$ images containing $n = 100^2$ pixels each. Let $0 \leq \lambda_1, \lambda_2, \lambda_3$ such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$. In a first step, we compute an approximation of the Wasserstein barycenter [2] with weights $\lambda = (\lambda_1, \lambda_2, \lambda_3)$ of the color of the first three images by solving the multi-marginal optimal transport problem in [37, Section 4.2]. We then transfer the color from the approximation of the barycenter onto the fourth image by solving a two-marginal optimal transport problem [267].

Let $x_i^{(\ell)} \in [0,1]^3$ denote the color of pixel i in image ℓ, where we treat the RGB values as element in $\mathbb{R}^3$ and rescale to $[0,1]^3$. Let $x_i^{(B)} = \lambda_1 x_i^{(1)} + \lambda_2 x_i^{(2)} + \lambda_3 x_i^{(3)}$ for $i = 1, \ldots, n$. We choose these points to define a reference template [335] for computing an approximation of the barycenter as in [37]. Let $\mathcal{C}_{i,j}^{(\ell,4)} = \|x_i^{(\ell)} - x_j^{(B)}\|_2^2$ for $1 \leq i,j \leq n$ and $\mathcal{K}^{(\ell,4)} = \exp(-\mathcal{C}^{(\ell,4)}/\eta)$ for $1 \leq \ell \leq 3$. We define the cost tensor

$$\mathcal{C}_{i,j,k,l} = \lambda_1 \mathcal{C}_{i,l}^{(1,4)} + \lambda_2 \mathcal{C}_{j,l}^{(2,4)} + \lambda_3 \mathcal{C}_{k,l}^{(3,4)} \quad \text{for } 1 \leq i,j,k,l \leq n,$$

and the Gibbs kernel tensor $\mathcal{K} = \exp(-\mathcal{C}/\eta)$ for a given regularization parameter $\eta \geq 0$. Following the ideas in [110, 150], we compute $r_B = r_4(\mathcal{P}_\eta^*)$, where

$$\mathcal{P}_\eta^* = \mathcal{K} \times_1 \operatorname{diag}(\exp(\beta_1)) \times_2 \operatorname{diag}(\exp(\beta_2)) \times_3 \operatorname{diag}(\exp(\beta_3)) \times_4 \operatorname{diag}(\mathbf{1}_n), \qquad (5.23)$$

with scaling parameters $\beta_1, \beta_2, \beta_3 \in \mathbb{R}^n$ chosen such that $r_\ell(\mathcal{P}_\eta^*) = \mathbf{1}$ for $1 \leq \ell \leq 3$. To compute an approximation $\mathcal{P}_\eta$ of $\mathcal{P}_\eta^*$ we run Algorithm 11 with cost tensor $\mathcal{C}$ and marginals $r_1 = r_2 = r_3 = \mathbf{1}_n$ and index selection (5.7), which we modify to use $\underset{\ell \in \{1,2,3\}}{\arg\max}$

instead of $\underset{\ell\in\{1,2,3,4\}}{\arg\max}$. This modification ensures that we only update the first three scaling parameters, which leads to an approximation of the form (5.23) [151, Theorem 3.5]. The Gibbs kernel corresponds to a star shaped graph structure as in Figure 5.2(b). The approximation of the color barycenter is now given by the points $x^{(B)}$ with masses r_B.

Remark 5.9. *Proposition 3.4 in [151] states that multi-marginal optimal transport problems with star shaped graph structures can be decomposed into several independent two-marginal problems when all marginals are prescribed. This does not apply for the computation of* (5.23)*, since* $r_4(\mathcal{P}_\eta^*)$ *is unknown.*

We now want to transfer the color from the approximation of the barycenter to the fourth image. We define the cost matrix $C_{i,j} = \|x_i^{(B)} - x_j^{(4)}\|_2^2$ for $1 \leq i, j \leq n$ and Gibbs kernel matrix $K = \exp(-C/\eta)$. Let the matrix P denote the approximate solution obtained from Algorithm 11 with cost matrix C and marginals $r_1 = r_B$ and $r_2 = \mathbf{1}_n$. The color vector of the target image with transferred colors is now given by $x_j^* = \sum_{i=1}^n P_{i,j} x_i^{(B)}$ for $1 \leq j \leq n$. Note that we can transfer the color to several target images without recomputing the barycenter approximation.

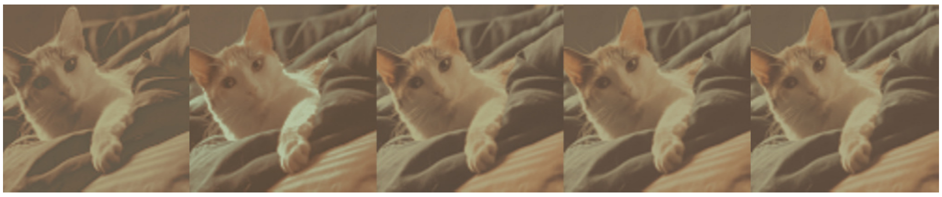
To accelerate the computation of marginals, we replace the Gibbs kernel tensor $\mathcal{K}$ by an approximation $\tilde{\mathcal{K}}$ obtained by replacing $\mathcal{K}^{(1,4)}, \mathcal{K}^{(2,4)}, \mathcal{K}^{(3,4)}$ by rank-r approximations using the randomized SVD [156]. We also replace the Gibbs kernel matrix K by a rank-r approximation $\tilde{K}$. Marginals and the target color vector x^* can now be computed in $\mathcal{O}(nr)$ operations.

In the following numerical experiments, we set $\eta = 1/10$ and compute approximate transport plans $\tilde{\mathcal{P}}, \tilde{P}$ by applying Algorithm 13 to $\tilde{\mathcal{K}}$ and $\tilde{K}$. In Figure 5.6, we study the impact of λ and r onto the color transferred image for example images from the COCO data set [218]. We observe that small values of r suffice to accurately approximate the desired image. The computation including the assembling of the matrices and the randomized SVD takes 0.25 seconds for $r = 50$, whereas using the full matrices in $\mathbb{R}^{10000\times 10000}$ directly in the tensor network takes 7.65 seconds, i.e. our low-rank approach reduces the computation time by over 96%.

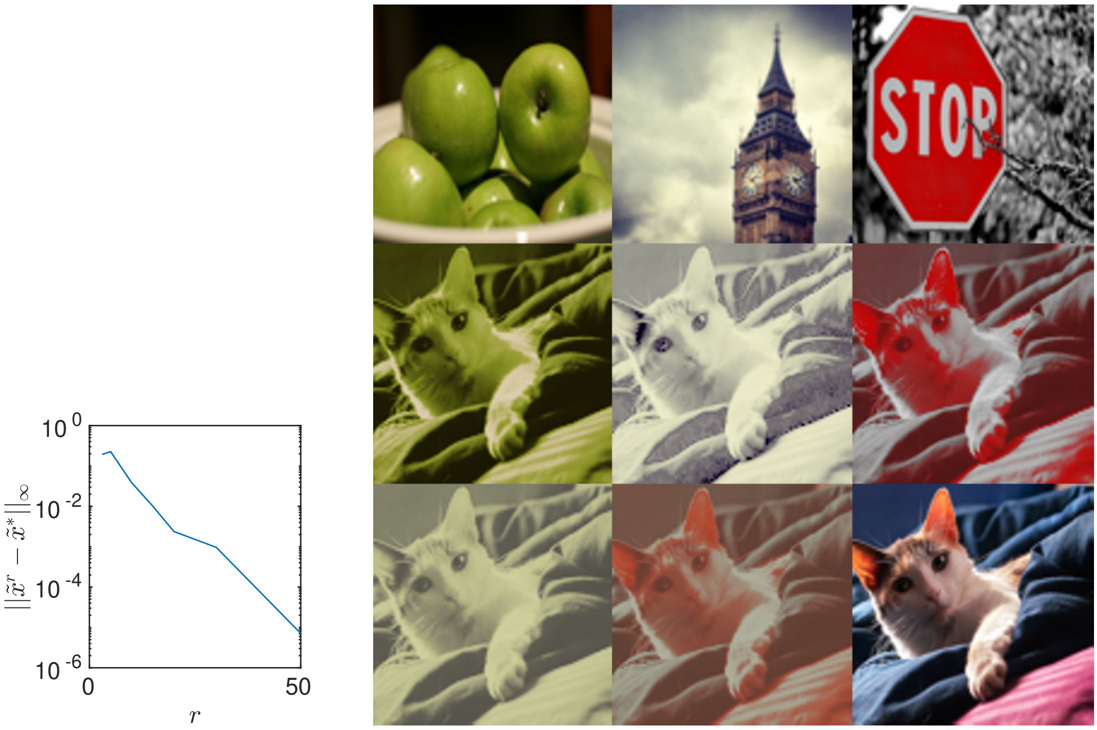
Remark 5.10. *When we are only interested in transferring the color onto a single target image, we can alternatively solve the following multi-marginal optimal transport problem without computing an approximation of the Barycenter. Based on the matrices* $\hat{\mathcal{C}}_{i,j}^{(\ell,4)} = \|x_i^{(\ell)} - x_j^{(4)}\|_2^2$ *for* $\ell = 1, 2, 3$*, we define the cost tensor*

$$\hat{\mathcal{C}}_{i,j,k,l} = \lambda_1 \hat{\mathcal{C}}_{i,l}^{(1,4)} + \lambda_2 \hat{\mathcal{C}}_{j,l}^{(2,4)} + \lambda_3 \hat{\mathcal{C}}_{k,l}^{(3,4)} \quad \textit{for } 1 \leq i, j, k, l \leq n.$$

Let $\hat{\mathcal{P}}$ *denote the transport plan obtained by solving the corresponding optimal transport problem with marginals* $r_\ell = \mathbf{1}_n$ *for* $\ell = 1, \ldots, 4$*. To accelerate the solution of this problem, we can again replace the matrices* $\exp(-\hat{\mathcal{C}}^{(1,4)}/\eta), \exp(-\hat{\mathcal{C}}^{(2,4)}/\eta), \exp(-\hat{\mathcal{C}}^{(3,4)}/\eta)$



(a) Impact of r

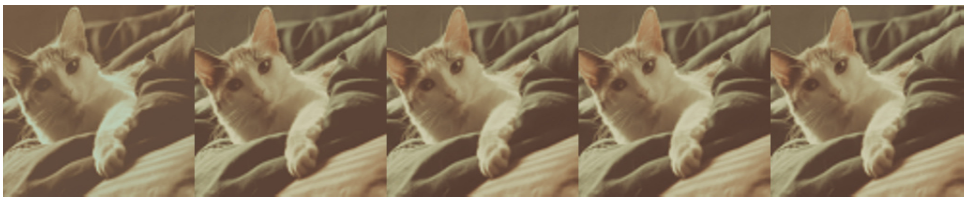


(b) Error decay

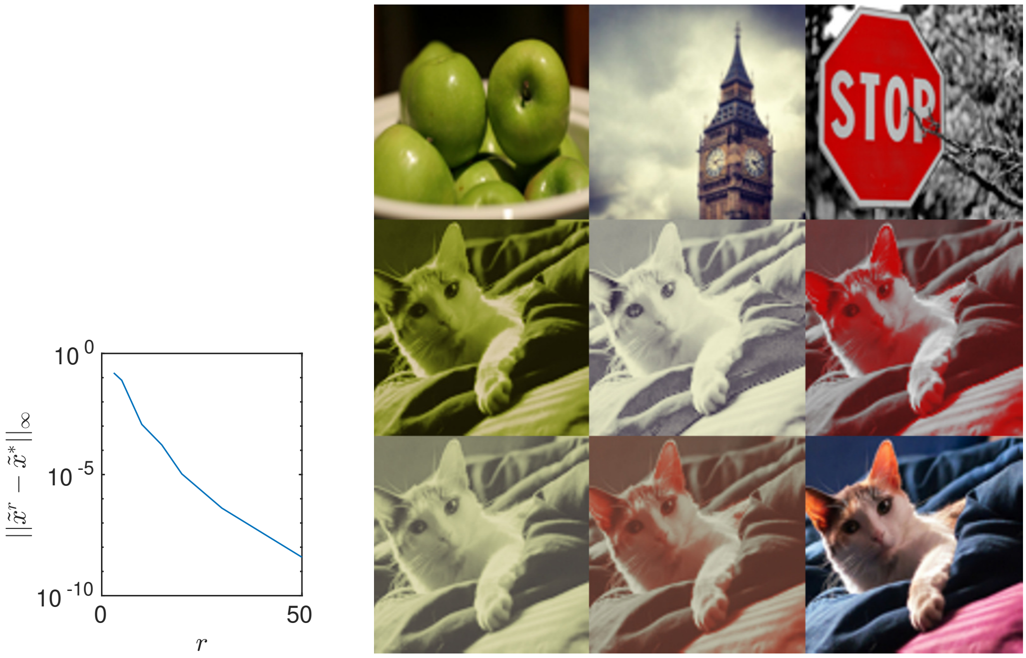
(c) Impact of λ

Figure 5.6 – Given the images displayed in the top row of (c), we use the method described in Section 5.5.2 to transfer their color to the bottom right image in (c). For fixed $\lambda = (1/3, 1/3, 1/3)$, we plot the resulting images for ranks $r = 3, r = 5, r = 10, r = 50$ from left to right in (a). The rightmost picture is obtained by using the full matrices $\mathcal{K}^\alpha$ and K directly. In (b), we plot $||\tilde{x}^r - \tilde{x}^*||_\infty$ where $\tilde{x}^r$ denotes the resulting image vector for a given rank r and $\tilde{x}^*$ is computed using the full matrices. Moreover, we display the resulting image for different values of λ in (c): middle row left to right $\lambda = (1, 0, 0)$, $\lambda = (0, 1, 0)$, $\lambda = (0, 0, 1)$, bottom row left $\lambda = (1/3, 2/3, 0)$, bottom row middle $\lambda = (1/5, 1/5, 3/5)$.

by rank-r approximations. Note that it would also be possible to decompose the problem into several independent two-marginal problems using Proposition 3.4 in [151]. The color vector of the target image with transferred colors can now be defined as $x_l^* = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n P_{i,j,k,l}(\lambda_1 x_i^{(1)} + \lambda_2 x_j^{(2)} + \lambda_3 x_k^{(3)})$ *for* $1 \leq l \leq n$. *In Figure 5.7, it is shown that the resulting target image seems to be similar to the image obtained after solving the Barycenter problem (see Figure 5.6). However, we actually solve different* *problems yielding different solutions. This is demonstrated in our experiments, where the maximum norm of the difference of the color vector obtained using this approach to the color vector obtained using the barycenter approach described in Section 5.5.2 is around* 94*. Also note that the error introduced by using rank-*r *approximations using this alternative approach is much smaller compared to using rank-*r *approximations using the barycenter approach.*



(a) Impact of r



(b) Error decay

(c) Impact of λ

Figure 5.7 – We repeat the experiments used to generate Figure 5.6 using the alternative approach described in Remark 5.10.

5.5.3 A tensor network with circles

In the following, we describe an optimal transport problem arising in the context of Schrödinger bridges [96, 151]. Let $m = 5$ and $n = 40^2$. We denote by $x^{(i)}$ for $1 \leq i \leq n$ the ith grid point on the grid $\{1, \dots, 40\}^2$. Let $C_{i,j} = ||x^{(i)} - x^{(j)}||_2^2$ for $1 \leq i, j \leq n$ and

$K = \exp(-C/\eta)$. We consider the Gibbs kernel

$$\mathcal{K}_{i_1,i_2,i_3,i_4,i_5} = \prod_{\alpha \in F} K_{I_\alpha} \quad \text{for } 1 \leq i_1, i_2, i_3, i_4, i_5 \leq n, \tag{5.24}$$

where $F = \{(1,2),(2,3),(3,4),(4,5)\}$. Note that this corresponds to a tensor network structure similar to Figure 5.2(a). Given $r_1, r_5 \in \Delta^n$, we now consider the problem of finding scaling parameters $\beta_1, \beta_5 \in \mathbb{R}^n$ such that

$$\mathcal{P}^*_\eta = \mathcal{K} \times_1 \operatorname{diag}(\exp(\beta_1)) \times_2 \operatorname{diag}(\mathbf{1}) \times_3 \operatorname{diag}(\mathbf{1}) \times_4 \operatorname{diag}(\mathbf{1}) \times_5 \operatorname{diag}(\exp(\beta_5)) \tag{5.25}$$

satisfies $r_1(\mathcal{P}^*_\eta) = r_1$ and $r_5(\mathcal{P}^*_\eta) = r_5$. In the context of Schrödinger bridges the marginals $r_\ell(\mathcal{P}^*_\eta)$ for $\ell = 2,3,4$ describe how the initial distribution r_1 most likely evolved into r_5 [151]. As in Section 5.5.2, we can again compute approximations of $\mathcal{P}^*_\eta$ by modifying Algorithm 13 such that only β_1 and β_5 are updated based on the prescribed marginals r_1, r_5; see [150, Section III.A].

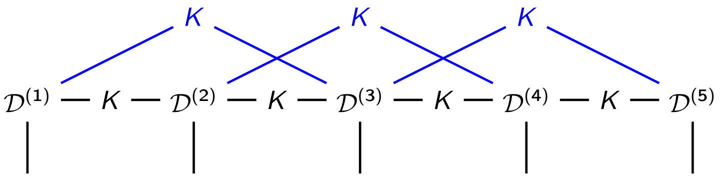

K
K
K
$\mathcal{D}^{(1)}$ — K — $\mathcal{D}^{(2)}$ — K — $\mathcal{D}^{(3)}$ — K — $\mathcal{D}^{(4)}$ — K — $\mathcal{D}^{(5)}$

Figure 5.8 – Tensor network diagram representation of $\mathcal{K}$ as defined in (5.24). The black network is obtained for $F = \{(1,2),(2,3),(3,4),(4,5)\}$. The blue part depicts the additional nodes and vertices introduced by setting $F = \{(1,2),(1,3),(2,3),(2,4),(3,4),(3,5),(4,5)\}$.

The Schrödinger bridge problem is based on a Markov chain model, in the sense that each distribution only depends on the previous distribution. We now introduce additional dependencies by setting $F = \{(1,2),(1,3),(2,3),(2,4),(3,4),(3,5),(4,5)\}$ in Equation (5.24). The tensor network structure of $\mathcal{K}$ is depicted in Figure 5.8. Note that this results in a graphical model for $\mathcal{P}^*_\eta$ with window graph structure as in [7, Figure 2]. In Figure 5.9, we depict the corresponding tensor network after replacing each matrix K by a rank-r approximation. Marginals of this network can be computed in $\mathcal{O}(nr^4)$ operations. For instance, to compute $r_3(\mathcal{P}^{(t)}_\eta)$ we first evaluate the colored tensors $\mathcal{T}^{[1]}, \ldots, \mathcal{T}^{[5]}$ in $\mathcal{O}(nr^4)$ operations by contracting the colored edges simultaneously using the structure of $\mathcal{D}^{(k)}$. We then compute $\mathcal{T}^{[1,2]} \in \mathbb{R}^{r\times r\times r}$ in $\mathcal{O}(r^4)$ operations by contracting $\mathcal{T}^{[1]}$ and $\mathcal{T}^{[2]}$ along their common edge. Analogously we compute $\mathcal{T}^{[4,5]}$. We then contract $\mathcal{T}^{[1,2]}$ and $\mathcal{T}^{[4,5]}$ along their common edge in $\mathcal{O}(r^5)$ operations before contracting all edges of the resulting tensor simultaneously with $\mathcal{T}^{[3]}$ in $\mathcal{O}(nr^4)$ operations. We want to emphasize that contracting the network without replacing K by low-rank approximations would not be feasible due to the required memory for storing the intermediate tensors.

In Figure 5.10, we study the impact of introducing these additional dependencies on the

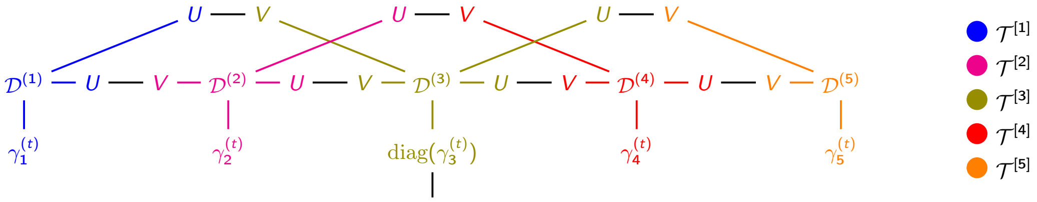

U — V
U — V
U — V
$\mathcal{D}^{(1)}$ — U — V — $\mathcal{D}^{(2)}$ — U — V — $\mathcal{D}^{(3)}$ — U — V — $\mathcal{D}^{(4)}$ — U — V — $\mathcal{D}^{(5)}$
$\gamma_1^{(t)}$
$\gamma_2^{(t)}$
$\mathrm{diag}(\gamma_3^{(t)})$
$\gamma_4^{(t)}$
$\gamma_5^{(t)}$
$\mathcal{T}^{[1]}$
$\mathcal{T}^{[2]}$
$\mathcal{T}^{[3]}$
$\mathcal{T}^{[4]}$
$\mathcal{T}^{[5]}$

Figure 5.9 – Tensor network diagram representation of $r_3(\mathcal{P}_\eta^{(t)})$ corresponding to the graph structure (5.24) with $F = \{(1,2),(1,3),(2,3),(2,4),(3,4),(3,5),(4,5)\}$. Here, we replace each matrix K by UV^T with $U, V \in \mathbb{R}^{n\times r}$. The colors mark subtensors $\mathcal{T}^{[1]}, \dots \mathcal{T}^{[5]}$ computed during the contraction.

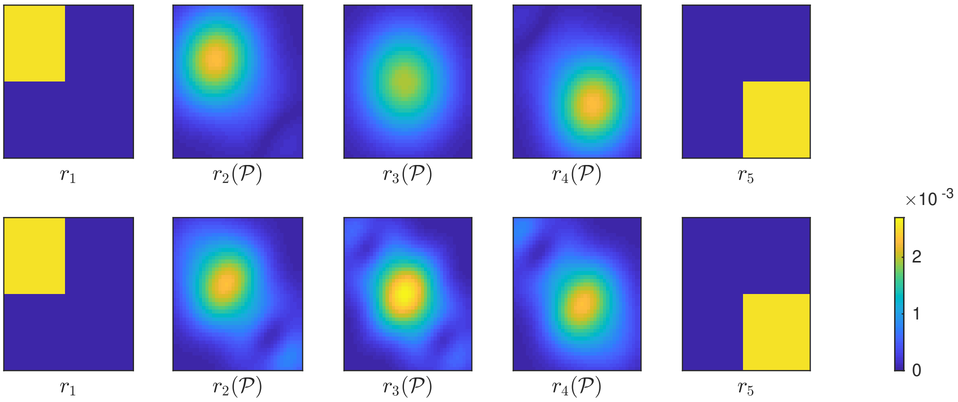


Figure 5.10 – We consider the scaling problem (5.25) for $\eta = 0.1$. We obtain an approximation $\tilde{\mathcal{K}}$ of the Gibbs kernel by replacing $\hat{\mathcal{K}}$ in (5.24) by a rank-10 approximation computed via the randomized SVD. We apply Algorithm 13 with index selection (5.7), where we take arg max only over the set $\{1,5\}$. Let $\mathcal{P}$ denote the resulting transport plan. In each row, we depict the prescribed r_1, r_5 as well as $r_2(\mathcal{P}), r_3(\mathcal{P}), r_4(\mathcal{P})$ reshaped into $\mathbb{R}^{40\times 40}$. The top row is obtained using $F = \{(1,2),(2,3),(3,4),(4,5)\}$ as in the Schrödinger bridge setting. The bottom row is obtained using $F = \{(1,2),(1,3),(2,3),(2,4),(3,4),(3,5),(4,5)\}$ with additional dependencies.

marginals $r_\ell(\mathcal{P})$, where $\mathcal{P}$ denotes the computed approximation of (5.25). We observe that the additional dependencies lead to a more concentrated $r_3(\mathcal{P})$, in the sense that the mass is less spread out. Moreover, the additional dependencies lead to $r_2(\mathcal{P})$ and $r_4(\mathcal{P})$ being concentrated slightly closer to the center of the images.

Remark 5.11. *In this experiment, the graphical model of the transport plan contains circles. Hence, we can not apply the belief propagation algorithm directly [118, 150]. There exist generalizations such as the loopy belief propagation algorithm [181], which does not always converge, and the junction tree algorithm [173], which applies the belief propagation on a so-called junction tree. This junction tree encodes the connection structure of the vertices. It can be used to determine the contraction order for the corresponding tensor network representation of the Gibbs kernel [273, Section 4.2]. In particular, the minimal cost to contract the tensor network is bounded from above by the cost of the junction tree* *algorithm.*

6 Self-diffusion matrix

This chapter is concerned with the computation of the self-diffusion matrix of a tagged particle process on a grid. In contrast to classical methods based on estimating long-time averages of empirical means of deviations of some stochastic processes on a finite-size supercell, we propose a novel numerical method based on solving the equivalent high-dimensional optimization problem formulated in [42, 210, 266] using an alternating linear scheme [39, 171, 274]. Once we obtained an approximation of the optimal solution, we use a carefully tuned variance reduction method to evaluate the desired entries of the self-diffusion matrix. To illustrate the method, we compute the self-diffusion matrix of a two-dimensional tagged particle process defined on a Cartesian grid. Our numerical experiments demonstrate that our novel low-rank approach is much less subject to statistical noise and that it yields a very accurate approximation of the self-diffusion matrix.

In addition, we present so-called cross-diffusion systems [49, 51, 182, 183] as example for applications, in which self-diffusion matrices play an important role. These systems appear in various fields such as population dynamics [290], tumor growth in medical biology [176], the modeling of biological systems [313] and lattice gases [13, 14, 111, 288], and the simulation of diffusion processes within mixtures of chemical compounds in materials science [23, 225]. When these systems are derived as hydrodynamic limits of lattice-based stochastic hopping models at the microscopic level [111, 266], they involve self-diffusion matrices as coefficients. Computing these self-diffusion coefficients is a major challenge that needs to be overcome before we can solve the cross-diffusion system numerically using for instance finite volume [116] or finite element methods [48].

This chapter is based on the articles [84, 85]. Its remainder is structured as follows. In Section 6.1, we recall the two equivalent definitions of the self-diffusion matrix of the tagged particle process. In Section 6.2, we construct finite-dimensional approximations of both definitions. The low-rank approach we propose here in order to compute a numerical approximation is presented in Section 6.3. The efficiency of our approach is illustrated through several numerical experiments presented in Section 6.4. Finally, in Section 6.5, we develop a cell-centered finite volume method to simulate a cross-diffusion system involving self-diffusion matrices as coefficients.

6.1 Infinite-dimensional definition

Let $d = 1, 2, 3$ denote the physical dimension of the problem. We consider a symmetric tagged particle process [276] defined on the infinite grid $\mathbb{Z}^d$. Let $K \in \mathbb{N}$ denote the number of possible jump directions for the particles. Let $(v_k)_{1\le k\le K} \subset \mathbb{Z}^d \setminus \{\mathbf{0}\}$ denote the set of possible jump directions. For all $1 \le k \le K$, the probability of jumping in the direction v_k is denoted by $p_k \in]0, 1]$. This jumping scheme is assumed to be symmetric in the sense that if the jump in the direction v_k occurs with probability p_k, then the jump in the direction $-v_k$ occurs with the same probability.

The self-diffusion matrix application

$$D_s : \begin{cases} [0,1] & \to & \mathbb{R}^{d\times d} \\ \rho & \mapsto & D_s(\rho) := (D_{s,ij}(\rho))_{1\le i,j\le d} \end{cases}$$

can be defined in the following two ways.

6.1.1 Definition as optimization problem

Let us first introduce some notation. Let $S := \mathbb{Z}^d \setminus \{\mathbf{0}\}$. For all so-called environments $\eta := (\eta_s)_{s\in S} \in \{0,1\}^S$ and all $y \neq z \in S$, we define by $\eta^{y,z} := (\eta_s^{y,z})_{s\in S}$ the element of $\{0,1\}^S$ such that

$$\eta_s^{y,z} := \begin{cases} \eta_s & \text{if } s \neq y, z, \\ \eta_y & \text{if } s = z, \\ \eta_z & \text{if } s = y. \end{cases}$$

Furthermore, for all $w \in S$, we define by $\eta^{\mathbf{0},w} := (\eta_s^{\mathbf{0},w})_{s\in S}$ the element of $\{0,1\}^S$ such that

$$\eta_s^{\mathbf{0},w} := \begin{cases} \eta_{s+w} & \text{if } s \neq -w, \\ 0 & \text{if } s = -w. \end{cases}$$

For a mean particle density $\rho \in [0, 1]$, we denote by $\rho^{\otimes}$ the Bernoulli product measure on $\{0,1\}^S$ with constant marginals equal to ρ in each mode. Let us also define the set of functions $H_\rho := L^2_{\rho^{\otimes}}\left(\{0,1\}^S\right)$, where $L^2_{\rho^{\otimes}}$ denotes the L^2 Lebesgue space with measure $\rho^{\otimes}$.

For a drift direction $u \in \mathbb{R}^d$ and mean particle density $\rho \in [0, 1]$, the self-diffusion coefficient $u^T D_s(\rho)\, u$ is given by:

$$u^T D_s(\rho)u := 2 \inf_{\Psi \in H_\rho} \mathbb{E}_{\rho^\otimes}\Big[\sum_{k=1}^{K} p_k \Big((1-\eta_{v_k}) \big(u \cdot v_k + \Psi(\eta^{\mathbf{0},v_k}) - \Psi(\eta)\big)^2 + \frac{1}{2} \sum_{\substack{y \in S \\ y+v_k \neq \mathbf{0}}} \big(\Psi(\eta^{y+v_k,y}) - \Psi(\eta)\big)^2 \Big)\Big], \tag{6.1}$$

where the notation $\mathbb{E}_{\rho^\otimes}$ refers to the fact that the expectation is taken over the product measure $\rho^\otimes$ [200, 266], i.e. we need to consider the expectation over all possible environments η. Problem (6.1) thus reads as an infinite-dimensional optimization problem over the set H_ρ.

Remark 6.1. *Naturally, for all $\rho \in [0,1]$, since $D_s(\rho)$ is a symmetric matrix, one can easily deduce the full matrix $D_s(\rho)$ from the knowledge of $u^T D_s(\rho)u$ for three vectors $u \in \mathbb{R}^d$.*

6.1.2 Definition as long time mean square deviation

The quantity $u^T D_s(\rho)u$ can be equivalently expressed as the long time limit of the following expectation [266, Theorem 2.3]. Let us assume that, at time $t = 0$, the tagged particle is located at position $\mathbf{0}$ and all other sites are occupied with probability ρ following a Bernoulli distribution. The duration between two consecutive jumping events follows an exponential law with parameter 1. The jumping directions (respectively rates) are given by $\{v_1, \dots, v_K\}$ (respectively $p_1, \dots, p_K$). Jumps are not allowed if the final site of the jumping particle is already occupied. Let $w(t)$ denote the position of the tagged particle at time $t \geq 0$. It then holds that, for a drift direction $u \in \mathbb{R}^d$ and mean particle density $\rho \in [0,1]$, the quantity $u^T D_s(\rho)u$ can be equivalently formulated as the long-time limit mean square deviation of the tagged particle in the direction $u \in \mathbb{R}^d$, i.e.

$$u^T D_s(\rho)u = 2 \lim_{t \to \infty} \frac{\mathbb{E}_{\rho^\otimes}[\langle u, w(t)\rangle^2]}{t}. \tag{6.2}$$

Remark 6.2. *Starting from a Bernoulli-product measure and symmetric transition rates, it is a classical problem in probability theory to study the motion of a tagged particle on $\mathbb{Z}^d$ [192, 215, 276] and more recently also on trees [68, 132]. For $d \geq 2$ and symmetric nearest neighbor transition rates, the tagged particle is known to satisfy a central limit theorem with non-degenerate limiting variance. However, to our best knowledge, a general closed formula for the limiting variance is not available.*

Notice that when starting from a Bernoulli-product measure with a tagged particle in the origin, the resulting environment process is stationary with respect to the Bernoulli-product measure conditioned to contain a particle in the origin, usually called the Palm measure. This suggests that the limiting variance for the tagged particle can be described in terms of *the sum of separable functions, i.e. in terms of a low-rank function. In turn, this indicates that the equivalent characterization of the limiting variance as in Equation* (6.1) *is also related to low-rank functions.*

6.2 Finite-dimensional approximation

The quantity $u^T D_s(\rho)u$ can not be computed exactly in practice, neither using expression (6.1) nor expression (6.2). On the one hand, (6.1) reads as an infinite-dimensional optimization problem and has to be approximated by a finite-dimensional optimization problem in practice [178, 210]. On the other hand, (6.2) requires the computation of the long-time average of the stochastic process defined on the infinite lattice grid $\mathbb{Z}^d$. Equivalently, it has to be approximated in a finite-dimensional setting, as long-time average of the mean-square deviation of the tagged particle associated to a stochastic process defined on a finite-size grid with periodic boundary conditions. Both finite-dimensional approximations are presented in detail in the following.

6.2.1 Discretized minimization problems

Let $M \in \mathbb{N}$ denote a discretization parameter and introduce the finite grid $S_M := \{-M, \cdots, M\}^d \setminus \{\mathbf{0}\}$. For the sake of simplicity, we assume that $M \geq \max_{k\in\{1,\dots,K\}} ||v_k||_1$. For any $\eta \in \{0,1\}^{S_M}$, we can construct by periodicity an extension $\widetilde{\eta} := (\widetilde{\eta}_s)_{s\in S} \in \{0,1\}^S$ by assuming with a slight abuse of notation that the site $\mathbf{0}$ is occupied, i.e. $\widetilde{\eta}_s = 1$ for $s \in (2M+1)\cdot\mathbb{Z}^d \setminus \{\mathbf{0}\}$. Using this notation, for all $\eta \in \{0,1\}^{S_M}$ and all $y, z, w \in S$, we define $\eta^{y,z} \in \{0,1\}^{S_M}$ and $\eta^{\mathbf{0},w} \in \{0,1\}^{S_M}$ as

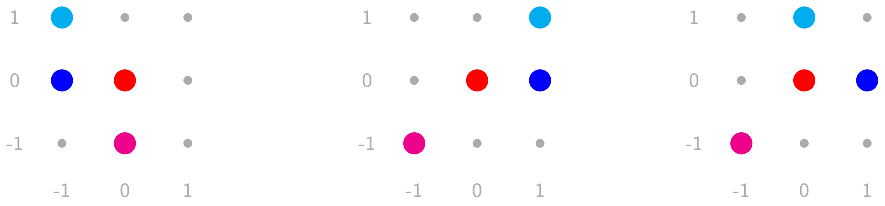
$$\eta^{y,z} := (\widetilde{\eta}_s^{y,z})_{s\in S_M} \qquad \text{and} \qquad \eta^{\mathbf{0},w} := \left(\widetilde{\eta}_s^{\mathbf{0},w}\right)_{s\in S_M}.$$

Figure 6.1 shows an illustration of $\eta^{y,z}$ and $\eta^{\mathbf{0},w}$.

Let $N := (2M+1)^d - 1 = \mathrm{Card}(S_M)$. For all $\ell \in \{0,\dots,N\}$, let $C_{M,\ell} := \{\eta \in \{0,1\}^{S_M} \,|\, \sum_{s\in S_M} \eta_s = \ell\}$ denote the set of all possible configurations of the particles on S_M so that the total number of occupied sites is equal to ℓ.

Let us define $H_M := \left\{\Psi : \{0,1\}^{S_M} \to \mathbb{R}\right\}$. For every $u \in \mathbb{R}^d$ and $l \in \{0,\dots,N\}$, we introduce the quadratic functional $A^u_{M,\ell} : H_M \to \mathbb{R}$ defined by

$$A^u_{M,\ell}(\Psi) := \frac{1}{|C_{M,\ell}|} \sum_{\eta\in C_{M,\ell}} \sum_{k=1}^{K} p_k \Big[(1-\eta_{v_k})\Big(u\cdot v_k + \Psi(\eta^{\mathbf{0},v_k}) - \Psi(\eta)\Big)^2 + \frac{1}{2} \sum_{\substack{y\in S_M\\ y+v_k\neq \mathbf{0}}} \left(\Psi(\eta^{y+v_k,y}) - \Psi(\eta)\right)^2\Big]. \tag{6.3}$$



$\eta^{\mathbf{0},(-1,0)}$ η $\eta^{(1,1)+(-1,0),(1,1)}$

Figure 6.1 – Middle: Visualization of one particular $\eta \in S_1$ for $d = 2$ with three occupied sites marked in blue, cyan and magenta. Additionally, we mark the tagged particle $\mathbf{0}$ in red. Left: Visualization of the occupied sites of $\eta^{\mathbf{0},(-1,0)}$. This can be seen as a jump of the imaginary red particle one step to the left, followed by immediately relabeling of the sites such that red particle remains at $\mathbf{0}$. By exploiting the periodicity, we obtain $\eta^{\mathbf{0},(-1,0)} \in S_1$. Right: Visualization of the occupied sites of $\eta^{(1,1)+(-1,0),(1,1)}$. This can be seen as jump of the cyan particle one step to the left.

Then, for some $\ell \in \{0, \dots, N\}$, one can define [42] for all $u \in \mathbb{R}^d$,

$$u^T D_s^M \left(\frac{\ell}{N}\right) u := 2 \min_{\Psi \in H_M} A^u_{M,\ell}(\Psi). \tag{6.4}$$

For any $\overline{\rho} \in [0, 1]$, it is proved in [210] that $\lim\limits_{\substack{M \to +\infty \\ \frac{\ell}{N} \to \overline{\rho}}} u^T D_s^M \left(\frac{\ell}{N}\right) u = u^T D_s(\overline{\rho}) u.$

6.2.2 Combined minimization problem

The collection of sets $C_{M,0}, \dots, C_{M,N}$ forms a partition of the set $\{0, 1\}^{S_M}$. Observe for a given $\Psi \in H_M$, $A^u_{M,\ell}(\Psi)$ only depends on the values of $\Psi(\eta)$ for $\eta \in C_{M,\ell}$, since $\eta \in C_{M,\ell}$ implies that also $\eta^{\mathbf{0},v_k} \in C_{M,\ell}$ and $\eta^{y+v_k,y} \in C_{M,\ell}$ for all $k \in \{1, \dots, K\}$. As a consequence, if $\Psi^{M,u}_{\mathsf{opt}} \in H_M$ is a minimizer of

$$\min_{\Psi \in H_M} A^u_M(\Psi), \tag{6.5}$$

where

$$A^u_M(\Psi) := \sum_{\eta \in \{0,1\}^{S_M}} \sum_{k=1}^{K} p_k \Big[(1 - \eta_{v_k}) \left(u \cdot v_k + \Psi(\eta^{\mathbf{0},v_k}) - \Psi(\eta)\right)^2 + \frac{1}{2} \sum_{\substack{y \in S_M \\ y + v_k \neq \mathbf{0}}} \left(\Psi(\eta^{y+v_k,y}) - \Psi(\eta)\right)^2 \Big], \tag{6.6}$$

then it holds that $A^u_{M,\ell}(\Psi^{M,u}_{\mathsf{opt}}) = \min_{\Psi \in H_M} A^u_{M,\ell}(\Psi)$ for all $\ell \in \{0, \ldots, N\}$. The knowledge of $\Psi^{M,u}_{\mathsf{opt}}$ then allows us to compute $u^T D^M_s\left(\frac{\ell}{N}\right)u$ for all $\ell = 0, \ldots, N$ as $2A^u_{M,\ell}(\Psi^{M,u}_{\mathsf{opt}})$. Note that the minimization problem (6.5) is independent of ℓ, in contrast to (6.4). Also note that the minimization problem (6.5) has 2^N degrees of freedom and is thus intractable for large values of N.

Our main contribution is to propose the low-rank tensor approximation algorithm to compute an approximation of $\Psi^{M,u}_{\mathrm{opt}}$ in order to mitigate this issue.

6.2.3 Estimation of long-time mean square deviation

The quantity $u^T D^M_s\left(\frac{\ell}{N}\right)u$ can be equivalently approximated by discretizing the definition (6.2). This leads to the problem of computing the long-time mean square deviation of a tagged particle evolving in the periodic environment S_M [210]. The Monte Carlo algorithm is the standard method of choice to compute an approximation of $u^T D^M_s\left(\frac{\ell}{N}\right)u$ by means of empirical averages. The quality of the obtained approximations then depends on the choice of two numerical parameters, namely the number of Monte Carlo samples (i.e. stochastic realizations of the periodized tagged particle process) and the value of the chosen final time.

We want to compare the low-rank tensor approximation algorithm we propose in this chapter with Monte Carlo estimations of the long-time mean square deviation. To this aim, we detail our Monte Carlo algorithm in the following. Let $\ell \in \{1, \ldots, N\}$. An initial environment η is obtained by sampling uniformly from $C_{M,\ell}$. We sample from exponential distributions to determine the next time a particle in the environment or the tagged particle performs a jump. Whenever a jump is performed the environment η is updated accordingly. Let $w = \mathbf{0} \in \mathbb{Z}^d$ denote the initial position of the tagged particle. Throughout the simulation, we track the position of the tagged particle in $\mathbb{Z}^d$, i.e. whenever the tagged particle successfully jumps in direction v_k, we set $w = w + v_k$. To approximate the expectation of w, we repeat this simulation N_s times with different samples. Each of these simulations is stopped at the same stopping time T at which we compute the approximation of $u^T D_s(\rho)u$. This approach is formalized in Algorithm 14.

Remark 6.3. *In Algorithm 14, we set $N_s = \lceil \hat{N}_s/(\ell+1) \rceil$, where $\hat{N}_s$ is a given input parameter. This ensures that the expected runtime is approximately equal for all choices of ℓ. Additionally Figure 6.5 shows that the variance of the output is comparable independently of ℓ. If we were to use the same N_s for all ℓ, we would observe a much larger variance for smaller ℓ and the runtime would increase for larger ℓ.*

Algorithm 14 Long-term Monte Carlo estimation

```
Input: $M \in \mathbb{N}$, $u \in \mathbb{R}^d$, number of occupied sites $1 \leq \ell \leq N$ with $N = (2M+1)^d - 1$, final time $T > 0$, $\hat{N}_s \in \mathbb{N}$
Output: approximation $\alpha \approx \frac{1}{2} u^T D_s^M \left(\frac{\ell}{N}\right) u = A^u_{M,\ell}(\Psi^{M,u}_{\mathsf{opt}})$
$w = \mathbf{0}$, $\alpha = 0$, $N_s = \lceil \hat{N}_s / (\ell + 1) \rceil$
for $i = 1, 2, 3, \dots, N_s$
    randomly initialize $\eta \in C_{M,\ell}$ and set $t_{\text{TOTAL}} = 0$,
    while true
        sample $t_{\text{NEW}}$ from an exponential distribution with mean $\frac{1}{\ell+1}$
        set $t_{\text{TOTAL}} = t_{\text{TOTAL}} + t_{\text{NEW}}$
        if $t_{\text{TOTAL}} > T$
            break
        uniformly select either one occupied site $y \in S_M$ with $\eta_y = 1$ or the tagged particle $y = \mathbf{0}$
        select a jump direction $v_k$ with probability $p_k$
        if $\tilde{\eta}_{y+v_k} = \mathbf{0}$, i.e. the target location is not occupied
            when $y \neq \mathbf{0} \Rightarrow$ set $\eta = \eta^{y+v_k,y}$
            when $y = \mathbf{0} \Rightarrow$ set $\eta = \eta^{\mathbf{0},v_k}$ and update $w = w + v_k$
    update $\alpha = \alpha + \langle u, w \rangle^2$
$\alpha = \frac{\alpha}{T N_s}$
```

6.3 Low-rank solutions for the optimization problem

The aim of this section is to present our novel low-rank method for the numerical resolution of the high-dimensional optimization problem (6.5). Instead of solving the problem for all functions in H_M, we restrict the solution space to functions of rank at most r defined analogous to (2.4), i.e. to functions of the form $\Psi = R^{(1)} + \cdots + R^{(r)}$, with

$$R(\eta)^{(k)} = \Pi_{s \in S_M} R_s(\eta_s)^{(k)}, \quad \forall \eta = (\eta_s)_{s \in S_M} \in \{0,1\}^{S_M},$$

for some $R_s^{(k)} : \{0,1\} \to \mathbb{R}$ for $s \in S_M$, $k = 1, \dots, r$. We denote by $H_M^r \subset H_M$ the set of all rank at most r functions. For all $r \in \mathbb{N}$, it holds

$$\min_{\Psi \in H_M} A^u_M(\Psi) = \min_{\Psi \in H_M^{2^N}} A^u_M(\Psi) \leq \min_{\Psi \in H_M^{r+1}} A^u_M(\Psi) \leq \min_{\Psi \in H_M^r} A^u_M(\Psi).$$

In the next sections, we derive an algorithm to approximate $\arg\min_{\Psi \in H_M^r} A^u_M(\Psi)$. We first introduce a fast and stable algorithm for the evaluation of A^u_M for functions in H^r_M. We then develop a successive minimization scheme to compute low-rank solutions of (6.5). Each minimization step is performed using an alternating linear scheme, which relies on the fast and stable evaluations of A^u_M. Lastly, we discuss the evaluation of $A^u_{M,\ell}$ for the computation of the self-diffusion coefficient (6.4).

6.3.1 Fast and stable evaluation

In this section, we introduce a fast and stable method to evaluate $A^u_M(\Psi)$ for $\Psi \in H^r_M$. The naive direct evaluation would require to sum over 2^N terms, which is intractable for large values of N. In principle, the order of summation can be exchanged leading to terms of the form $\sum_{\eta\in\{0,1\}^{S_M}} \Psi(\eta)^2$, which can be evaluated efficiently for $\Psi \in H^r_M$. However, subtracting these terms leads to a lot of numerical cancellation for $M > 1$. We circumvent these issues by treating the evaluation of $A^u_M(\Psi)$ as the computation of the Frobenius norms of certain tensors. These Frobenius norms can then be evaluated in a fast and stable way.

Reformulation as tensor norm

For $\Psi \in H_M$, we consider the order N tensor $\mathcal{T}^{(\Psi)} \in \mathbb{R}^{2\times\cdots\times 2}$ with entries $\mathcal{T}^{(\Psi)}_{i_1,\dots,i_N} = \Psi(\eta)$, where $\eta = (\eta_{s_1}, \dots, \eta_{s_N})$ for some enumeration $s_1, \dots, s_N$ of the sites in S_M and $\eta_{s_j} = i_j - 1$ for $1 \le j \le N$. Note that $\Psi \in H^r_M$, as defined in Section 6.3, can be represented by a rank-r tensor in CP-format (2.4) of the form

$$\mathcal{T}^{(\Psi)}_{i_1,\dots,i_N} = \sum_{k=1}^{r} \prod_{j=1}^{N} a^{(j,k)}_{i_j}, \tag{6.7}$$

where $a^{(j,k)} \in \mathbb{R}^2$ is defined as $a^{(j,k)} = (R^{(k)}_{s_j}(0), R^{(k)}_{s_j}(1))$.

For $v, y \in S_M$, we analogously define the order N tensor $\mathcal{T}^{(\Psi^{(\mathbf{0},v)})} \in \mathbb{R}^{2\times\cdots\times 2}$ with entries $\mathcal{T}^{(\Psi^{(\mathbf{0},v)})}_{i_1,\dots,i_N} = \Psi(\eta^{\mathbf{0},v})$. When $y + v \neq \mathbf{0}$ we additionally define the order N tensor $\mathcal{T}^{(\Psi^{(y+v,y)})} \in \mathbb{R}^{2\times\cdots\times 2}$ with entries $\mathcal{T}^{(\Psi^{(y+v,y)})}_{i_1,\dots,i_N} = \Psi(\eta^{y+v,y})$. We observe that these again are rank-r tensors when $\Psi \in H^r_M$. Lastly, we define the order N, rank-1 tensor $\mathcal{T}^{(u\cdot v)} \in \mathbb{R}^{2\times\cdots\times 2}$ with entries $\mathcal{T}^{(u\cdot v)}_{i_1,\dots,i_N} = u \cdot v$.

For an order N tensor $\mathcal{T} \in \mathbb{R}^{2\times\cdots\times 2}$ and a site s we define the projection operators $\mathcal{P}_s : \mathbb{R}^{2\times\cdots\times 2} \to \mathbb{R}^{2\times\cdots\times 2}$ as

$$(\mathcal{P}_s(\mathcal{T}))_{i_1,\dots,i_N} := \begin{cases} \mathcal{T}_{i_1,\dots,i_N} & i_s = 1 \\ 0 & \text{otherwise} \end{cases}, \quad \text{for } 1 \le i_1, \dots, i_N \le 2,$$

where i_s denotes to the index assigned to site s. We can now rewrite (6.6) as

$$A^u_M(\Psi) = \sum_{k=1}^{K} p_k \Bigg[\|P_{v_k}\Big(\mathcal{T}^{(u\cdot v_k)} + \mathcal{T}^{(\Psi^{(\mathbf{0},v_k)})} - \mathcal{T}^{(\Psi)}\Big)\|_F^2 + \frac{1}{2} \sum_{\substack{y\in S_M \\ y+v_k\neq\mathbf{0}}} \|\mathcal{T}^{(\Psi)} - \mathcal{T}^{(\Psi^{(y+v_k,y)})}\|_F^2 \Bigg]. \tag{6.8}$$

Efficient evaluation of Frobenius norms for sums of low-rank tensors

Let $k \in \{1, \dots, K\}$ and $y \in S_M$ such that $y + v_k \neq \mathbf{0}$ be fixed. We derive an algorithm inspired by TT orthogonalization [249] to evaluate $\|\mathcal{T}^{(\Psi)} - \mathcal{T}^{(\Psi^{(y+v_k,y)})}\|_F^2$.

We start by rewriting Equation (6.7) in TT format (2.6) as

$$\mathcal{T}^{(\Psi)}_{i_1,\dots,i_N} = \sum_{k_1=1}^{r} \cdots \sum_{k_{N-1}=1}^{r} \mathcal{C}^1_{1,i_1,k_1} \mathcal{C}^2_{k_1,i_2,k_2} \mathcal{C}^3_{k_2,i_3,k_3} \cdots \mathcal{C}^N_{k_{N-1},i_N,1}, \tag{6.9}$$

with tensors $\mathcal{C}^1 \in \mathbb{R}^{1\times 2\times r}$, $\mathcal{C}^N \in \mathbb{R}^{r\times 2\times 1}$ and $\mathcal{C}^j \in \mathbb{R}^{r\times 2\times r}$ for $1 < j < N$, whose entries are given by

$$\begin{aligned}
&\mathcal{C}^j_{k_1,i,k_2} = \begin{cases} a_i^{j,k_1} & k_1 = k_2 \\ 0 & \text{otherwise} \end{cases}, && \text{for } 1 < j < N,\ 1 \leq i \leq 2,\ 1 \leq k_1, k_2 \leq N, \\
&\mathcal{C}^1_{1,i,k} = a_i^{1,k} && \text{for } 1 \leq i \leq 2,\ 1 \leq k \leq N, \\
&\mathcal{C}^N_{k,i,1} = a_i^{N,k} && \text{for } 1 \leq i \leq 2,\ 1 \leq k \leq N.
\end{aligned}$$

The representation (6.9) can be generalized to other low-rank tensors. Let $\mathcal{T}^{(\Psi^{(y+v_k,y)})}$ be analogously represented by tensors $\mathcal{D}^j$. The tensor $\mathcal{T}^{(\Psi)} - \mathcal{T}^{(\Psi^{(y+v_k,y)})}$ is at most rank-$2r$. It's representation in the form of (2.6) with tensors $\mathcal{E}^j$ can be constructed from the tensors $\mathcal{C}^j, \mathcal{D}^j$, i.e. the tensors $\mathcal{E}^1 \in \mathbb{R}^{1\times 2\times 2r}$, $\mathcal{E}^N \in \mathbb{R}^{2r\times 2\times 1}$ and $\mathcal{E}^j \in \mathbb{R}^{2r\times 2\times 2r}$ for $1 < j < N$ are defined as $\mathcal{E}^1_{1,:,1:r} = \mathcal{C}^1$, $\mathcal{E}^1_{1,:,r+1:2r} = -\mathcal{D}^1$, $\mathcal{E}^N_{1:r,:,1} = \mathcal{C}^N$, $\mathcal{E}^N_{r+1:2r,:,1} = \mathcal{D}^N$ and $\mathcal{E}^j_{1:r,:,1:r} = \mathcal{C}^j$, $\mathcal{E}^j_{r+1:2r,:,r+1:2r} = \mathcal{D}^j$. Note that for $1 < j < N$ these tensors are sparse by construction.

We can compute the norm $\|\mathcal{T}^{(\Psi)} - \mathcal{T}^{(\Psi^{(y+v_k,y)})}\|_F^2$ directly form the tensors $\mathcal{E}^j$ in a fast and stable way using TT orthogonalization [249] as defined in Algorithm 15. We want to emphasize that the sparsity structure of $\mathcal{E}^j$ is preserved in Algorithm 15. The evaluation of $\|\mathcal{E}^N\|_F^2$ in line 13 requires summing up $4r$ terms, whereas a naive evaluation of $\|\mathcal{T}^{(\Psi)} - \mathcal{T}^{(\Psi^{(y+v_k,y)})}\|_F^2$ involves 2^N terms.

Remark 6.4. *The sum of several low-rank tensors can be represented in the format* (2.6). *The representation of the sum can again be constructed from the representations of the individual low-rank tensors. This allows us to apply TT orthogonalization to evaluate all Frobenius-norms in Equation* (6.8) *in a fast and stable way.*

6.3.2 Successive minimization

Let $r \in \mathbb{N}$. In order to approximate $\underset{\Psi \in H_M^r}{\arg\min}\, A_M^u(\Psi)$, we decompose the problem into a sequence of successive minimization problems following the idea of Remark 2.3. We assume that $\Psi \in H_M^r$ is of the form $\Psi = R^{(1)} + \cdots + R^{(r)}$ with rank-1 functions $R^{(k)} \in H_M^1$.

Algorithm 15 Frobenius norm evaluation

```
Input: tensors $\mathcal{E}^1 \in \mathbb{R}^{1\times 2\times 2r}$, $\mathcal{E}^N \in \mathbb{R}^{2r\times 2\times 1}$ and $\mathcal{E}^j \in \mathbb{R}^{2r\times 2\times 2r}$ for $1 < j < N$
Output: $\|\mathcal{T}\|_F^2$, where $\mathcal{T} \in \mathbb{R}^{2^N}$ has entries $T_{i_1,\dots,i_N} = \sum_{k_1=1}^{r} \cdots \sum_{k_{N-1}=1}^{r} \mathcal{E}^1_{1,i_1,k_1}\mathcal{E}^2_{k_1,i_2,k_2}\cdots\mathcal{E}^N_{k_{N-1},i_N,1}$
Compute $QR$ decomposition of $\mathcal{E}^1$ reshaped as element in $\mathbb{R}^{2\times 2r}$.
Set $\mathcal{E}^1$ to $Q$ reshaped as element in $\mathbb{R}^{1\times 2\times 2r}$.
Update $\mathcal{E}^2$: reshape to $\mathbb{R}^{2r\times 2\cdot 2r}$, multiply with $R$ from the left, reshape back to $\mathbb{R}^{2r\times 2\times 2r}$.
for $j = 2,\dots,N-1$
    Compute $QR$ decomposition of $\mathcal{E}^j$ reshaped as element in $\mathbb{R}^{2r\cdot 2\times 2r}$.
    Set $\mathcal{E}^j$ to $Q$ reshaped as element in $\mathbb{R}^{2r\times 2\times 2r}$.
    if $j < N-1$
        Update $\mathcal{E}^{j+1}$: reshape to $\mathbb{R}^{2r\times 2\cdot 2r}$, multiply with $R$ from the left, reshape back to $\mathbb{R}^{2r\times 2\times 2r}$.
    else
        Update $\mathcal{E}^N$: reshape to $\mathbb{R}^{2r\times 2}$, multiply with $R$ from the left, reshape back to $\mathbb{R}^{2r\times 2\times 1}$.
$\|\mathcal{T}\|_F^2 = \|\mathcal{E}^N\|_F^2$
```

We first determine $R^{(1)}$ as solution of $\min_{R^{(1)}\in H^1_M} A^u_M(R^{(1)})$. In a successive step $R^{(2)}$ is determined as solution of $\min_{R^{(2)}\in H^1_M} A^u_M(R^{(1)} + R^{(2)})$. This is continued successively until Ψ is determined. The idea is formalized in Algorithm 16.

Algorithm 16 Successive minimization

```
Input: rank $r$
Output: approximation $\Psi \approx \arg\min_{\Psi\in H^r_M} A^u_M(\Psi)$
$\Psi = 0$
for $k = 1,\dots,r$
    $R^{(k)} = \arg\min_{R\in H^1_M} A^u_M(\Psi + R)$
    $\Psi = \sum_{i=1}^{k} R^{(i)}$
```

6.3.3 Alternating least squares

In Algorithm 16, we need to solve minimization problems of the form

$$\min_{R\in H^1_M} A^u_M(\Psi + R) \tag{6.10}$$

for given $\Psi \in H^r_M$. In the following, we introduce an ALS algorithm (see Section 2.1.2) to solve such minimization problems. The main idea is to approximate the solution of (6.10) by an iterative scheme which amounts to solving a sequence of small-dimensional linear problems. We start from an initial $R(\eta) := \Pi_{s\in S_M} R_s(\eta_s)$. We first minimize $A^u(\Psi + R)$ only with respect to a selected $R_{s_0} : \{0,1\} \to \mathbb{R}$ for some $s_0 \in S_M$ leaving the other R_s,

$s \neq s_0$ fixed. By partially evaluating $A^u(\Psi + R)$ for all terms not depending on R_{s_0}, we obtain that $\min_{R_{s_0} \in \{\{0,1\} \to \mathbb{R}\}} A^u(\Psi + R)$ with $R(\eta) := R_{s_0}(\eta_{s_0}) \Pi_{s \in S_M \setminus \{s_0\}} R_s(\eta_s)$ is equivalent to a quadratic optimization problem

$$\min_{R_{s_0} \in \{\{0,1\} \to \mathbb{R}\}} \alpha_1 R_{s_0}(1)^2 + \alpha_2 R_{s_0}(0)^2 + \alpha_3 R_{s_0}(1) R_{s_0}(0) + \alpha_4 R_{s_0}(1) + \alpha_5 R_{s_0}(0) + \alpha_6, \quad (6.11)$$

with constants $\alpha_1, \dots, \alpha_6 \in \mathbb{R}$ depending on the fixed R_s, $s \neq s_0$ and Ψ. This quadratic optimization problem always admits a unique optimal R_{s_0}, which is given by $R_{s_0}(1) = a$ and $R_{s_0}(0) = b$, where $a, b \in R$ are the solution of the linear system

$$\begin{pmatrix} 2\alpha_1 & \alpha_3 \\ \alpha_3 & 2\alpha_2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} -\alpha_4 \\ -\alpha_5 \end{pmatrix}. \quad (6.12)$$

This allows us to optimize $A^u_M(\Psi + R)$ with respect to individual R_{s_0}. By alternating the selected $s_0 \in S_M$, we obtain the alternating least squares algorithm, which is formalized in Algorithm 17.

Algorithm 17 Alternating least squares

```
Input: initial functions $R_s^0 : \{0,1\} \to \mathbb{R}$ for $s \in S_M$, function $\Psi \in H^r_M$, vector $u$, tolerance $\varepsilon$
Output: approximation $R_{\text{opt}}(\eta) = \Pi_{s \in S_M} R_s(\eta_s)$ of $\arg\min_{R \in H^1_M} A^u_M(\Psi + R)$
$v_{\mathsf{old}} = \infty$, $v_{\mathsf{new}} = A^u_M(\Psi + R^0)$ with $R^0(\eta) := \Pi_{s \in S_M} R^0_s(\eta_s)$
$\forall s \in S_M,\ R_s := R^0_s$.
while $|v_{\mathsf{old}} - v_{\mathsf{new}}| > \varepsilon |v_{\mathsf{new}}|$.
    $v_{\mathsf{old}} = v_{\mathsf{new}}$
    for $s_0 \in S_M$
        $R_{s_0} = \arg\min_{\widetilde{R}_{s_0} : \{0,1\} \to \mathbb{R}} A^u(\Psi + \widetilde{R})$ where $\widetilde{R}(\eta) = \widetilde{R}_{s_0}(\eta_{s_0}) \Pi_{s \in S_M \setminus \{s_0\}} R_s(\eta_s)$ for all $\eta = (\eta_s)_{s \in S_M}$
    $v_{\mathsf{new}} = A^u_M(\Psi + R)$
```

Remark 6.5. *To compute the constants* α_i*, we can either explicitly implement the partial evaluations of* $A^u_M(\Psi + R)$*. Alternatively, we can treat* $A^u_M(\Psi + R)$ *as a function in* $\mathbb{R}^2 \to \mathbb{R}$ *depending on the values* $R_{s_0}(0)$ *and* $R_{s_0}(1)$*. We know that this function is a multivariate-polynomial of the form* $\alpha_1 R_{s_0}(1)^2 + \alpha_2 R_{s_0}(0)^2 + \alpha_3 R_{s_0}(1) R_j(0) + \alpha_4 R_{s_0}(1) + \alpha_5 R_{s_0}(0) + \alpha_6$*. The constants can be computed using multivariate-polynomial interpolation* (2.16) *in six points. This interpolation has the advantages that it is non-intrusive and that the evaluations of* $A^u_M(\Psi + R)$ *can be performed efficiently using the ideas of Section 6.3.1.*

Remark 6.6. *Throughout this chapter, we use approximations in the CP-format. Note that alternating algorithms can also be applied to TT tensors [145, 171, 312].*

6.3.4 Monte Carlo methods

Let $\Psi \in H_M^r$ denote an approximation of the solution of (6.5). In the following, we discuss how to evaluate $A_{M,\ell}^u(\Psi)$. In the definition (6.3) of $A_{M,\ell}^u$ the function

$$f(\eta) := \sum_{k=1}^{K} p_k \Bigg[(1-\eta_{v_k}) \left(u \cdot v_k + \Psi(\eta^{\mathbf{0},v_k}) - \Psi(\eta) \right)^2 + \frac{1}{2} \sum_{\substack{y \in S\setminus\{\mathbf{0}\} \\ y+v_k \neq \mathbf{0}}} \left(\Psi(\eta^{y+v_k,y}) - \Psi(\eta) \right)^2 \Bigg] \tag{6.13}$$

is evaluated for all $\eta \in C_{M,\ell}$. For larger N and most choices of ℓ, evaluating $|C_{M,\ell}| = \binom{N}{\ell}$ terms is intractable. We thus propose to use a Monte Carlo method [227] to approximate $A_{M,\ell}^u(\Psi)$.

In a naive Monte Carlo method, we compute N_s samples $\eta^{(i)} \in C_{M,\ell}$ for $i = 1, \ldots, N_s$ and replace $\frac{1}{|C_{M,\ell}|} \sum_{\eta \in C_{M,\ell}}$ in (6.3) by $\frac{1}{N_s} \sum_{\eta \in \{\eta^{(1)}, \ldots, \eta^{(N_s)}\}}$.

In Algorithm 18, we describe a Monte Carlo method with additional variance reduction [158], which as demonstrated in Section 6.4 reduces the number of samples needed to obtain a given approximation accuracy. The main idea is to observe that the sites $v_1, \ldots, v_K$ play a special role since they are the most relevant for jumps of the tagged particle. Instead of sampling the η uniformly in $C_{M,\ell}$, we now ensure that all possible states of the sites $v_1, \ldots, v_K$ are occurring equally often in the set of sample points. We obtain the environments used to approximate $A_{M,\ell}^u(\Psi)$ using the following construction. We first sample the states of all sites other than $v_1, \ldots, v_K$ randomly. We then combine these states with all possible states of the sites $v_1, \ldots, v_K$ to obtain 2^K different environments. After evaluating f for each of these environments, we obtain the next set of environments by randomly sampling the states of all sites other than $v_1, \ldots, v_K$. This procedure is repeated until desired the total number of evaluations of f has been reached.

Remark 6.7. *In line 9 of Algorithm 18, we sample form the set set*

$$\{\vec{w}^{(2)} \in \{0,1\}^{S_M \setminus \{v_1, \ldots, v_k\}} \mid ||\vec{w}^{(2)}||_1 = n_2\}$$

which contains $\binom{N-K}{n_2}$ *elements. When the number of elements in this set is smaller than* $\widetilde{N}_s$*, we can compute* $\tilde{S}$ *based on all possible* $\vec{w}^{(2)}$ *instead of sampling* $\widetilde{N}_s$ *times. This decreases the variance of the approximation further and reduces the number of required evaluations of* f*.*

6.3.5 Limitations of the approach

In this section, we briefly discuss the limitations of the proposed algorithm. These are

Algorithm 18 Monte Carlo method with variance reduction

```
Input: function $f : C_{M,\ell} \to \mathbb{R}$, parameter $\widetilde{N}_s$
Output: approximation $S$ of $\frac{1}{|C_{M,\ell}|} \sum_{\eta \in C_{M,\ell}} f(\eta)$
$S = 0$
for $\vec{w}^{(1)} \in \{0,1\}^K$
    $n_1 := ||\vec{w}||_1$, $n_2 := \ell - n_1$
    if $0 \leq n_2 \leq N - \ell$
        $\tilde{S} = 0$
        for $i = 1, \dots, \widetilde{N}_s$
            Sample $\vec{w}^{(2)} \in \{0,1\}^{S_M \setminus \{v_1, \dots, v_k\}}$ such that $||\vec{w}^{(2)}||_1 = n_2$.
            Construct $\eta \in C_{M,\ell}$ such that $\eta(s) = \begin{cases} \vec{w}^{(1)}_k & s = v_k \\ \vec{w}^{(2)}_s & \text{otherwise} \end{cases}$.
            $\tilde{S} = \tilde{S} + f(\eta)$
        $S = S + \binom{N-K}{n_2} \cdot \tilde{S} / \widetilde{N}_s$
$S = \frac{S}{\binom{N}{\ell}}$
```

primarily related to the following observation. Since $A^u_{M,\ell}(\Psi^{M,u}_{\mathsf{opt}})$ defines the entries of the self-diffusion coefficient (6.4), we know that $A^u_{M,\ell}(\Psi^{M,u}_{\mathsf{opt}}) \in [0,1]$. Note that $A^u_{M,\ell}$ contains the normalization constant $|C_{M,\ell}|^{-1}$. The objective function A^u_M (6.6) does not include a normalization factor. This implies that the $A^u_M(\Psi^{M,u}_{\mathsf{opt}})$ is of order 2^N. Trying to minimize this objective function numerically using floating point arithmetic leads to issues caused by rounding errors for large N.

For larger values of the objective function, it becomes increasingly challenging to solve the individual minimization problems in line 8 of Algorithm 17. In particular, our approach of using polynomial interpolation to find the location of the minimum might encounter numerical rounding issues. We find that the positive eigenvalues of the 2×2 matrix in (6.12) tend to be many orders of magnitude smaller than the norm of the right hand side. This implies that small rounding errors in the constants $\alpha_1, \dots, \alpha_6$ might lead to vastly different solutions. For $M > 2$, we even find that rounding errors can lead to negative eigenvalues in the system, i.e. we can not find the minimum of the polynomial (6.11) at all. It might be necessary to use a different approach to solve the individual ALS minimization problems for larger M. Alternatively, one could develop an algorithm to minimize the function $\log(A^u_M)$.

Moreover, the number of ALS iterations needed to find a good rank-1 minimizer tends to increase when increasing the size of the domain. This is especially true when a random initialization is used in Algorithm 17. Note that it might be possible to circumvent this issue by initializing Algorithm 17 based on the solution computed for a smaller value of M. Overcoming these limitations will then require further investigation. A possible path could be to combine domain decomposition approaches together with the tensor-based optimization algorithm we propose here, in order to obtain a global optimization procedure where only independent parallel local optimization problems on medium-sized cells are solved.

6.4 Numerical Experiments

In the following numerical experiments[*], we consider a tagged particle process with $K = 4$ displacement vectors $v_1 = (1, 0)$, $v_2 = (-1, 0)$, $v_3 = (0, 1)$, and $v_4 = (0, -1)$ and with associated probability $p_k = 1/4$ for $k = 1, 2, 3, 4$. All computing times are measured without parallelization in MATLAB R2018b on a Lenovo Thinkpad T480s with Intel Core i7-8650U CPU and 15.4 GiB RAM. Note that both the sampling Algorithm 14 and Algorithm 18 as well as the evaluation of the different Frobenius-norms in Equation (6.8) can be parallelized to speed up computations.

6.4.1 Solving the optimization problem

In the following, we analyze the numerical approximation of the self-diffusion coefficient by solving the optimization problem (6.4).

Low-rank approximation error First, we study how the rank r affects the value of $\min_{\Psi \in H_M^r} A_M^u(\Psi)$ for $u = (1, 0)$. Let $\Psi_{\mathsf{ALS}}^{r,u} \in H_M^r$ denote the function obtained by using Algorithm 16 with rank r, where the minimization problem in each iteration is solved using Algorithm 17. The initial functions R_s^0 in Algorithm 17 are selected randomly by assigning random values drawn form the uniform distribution on $[0, 1]$ to $R_s^0(0)$ and $R_s^0(1)$. In Algorithm 17, we set $\varepsilon = 10^{-12}$ and additionally stop after line 8 has been executed 420 times.

We depict the error $\|A_1^u(\Psi_{\mathsf{ALS}}^{r,u}) - A_1^u(\Psi_{\mathsf{ALS}}^{10,u})\|/\|A_1^u(\Psi_{\mathsf{ALS}}^{10,u})\|$, evaluated using the ideas of Section 6.3.1, of $\Psi_{\mathsf{ALS}}^{r,u}$ for various r in Figure 6.2. The computation with $r = 10$ takes 4 minutes for $M = 1$ and 82 minutes for $M = 2$. We observe that the error decays quickly with increasing rank. In particular, for $M = 1$ an approximation with $r = 1$ is less than 0.1% away from a direct least squares solution of the minimization problem (6.5) (see Remark 6.8).

Remark 6.8. *Given the tensor* $\mathcal{T}^{(\Psi)}$ *as defined in* (6.7), *we can equivalently write the evaluation of* $A_M^u(\Psi)$ *as sum of* $Q = NK \cdot 2^{N-1}$ *quadratic terms, which can be expressed as* $\|B\mathsf{vec}(\mathcal{T}^{(\Psi)}) + b\|_2^2$ *for some matrix* $B \in \mathbb{R}^{Q \times 2^N}$ *and vector* $b \in \mathbb{R}^Q$*. Thus, (6.5) boils down to solving the least squares problem*

$$\min_{\Psi \in H_M} \|B\mathsf{vec}(\mathcal{T}^{(\Psi)}) + b\|_2^2. \tag{6.14}$$

---

[*] The code to reproduce these results is available from https://github.com/cstroessner/SelfDiffusion

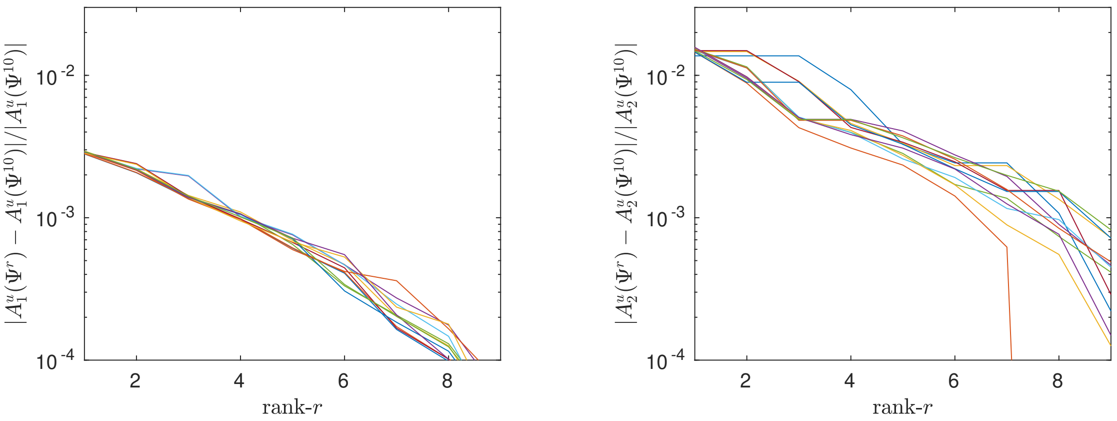


Figure 6.2 – Algorithm 16 yields successive approximations $\Psi_{\mathsf{ALS}}^{r,u} \in H_M^r$. For $r \in \{1, \dots, 9\}$, we plot the relative error of $\Psi_{\mathsf{ALS}}^{k,u}$ compared to $\Psi_{\mathsf{ALS}}^{10,u}$. We repeat this experiment 12 times with different random initial functions in Algorithm 17. The resulting relative errors are displayed in different colors. Left: $M = 1$. Right: $M = 2$.

In our setting of $d = 2$, $M = 1$ and $K = 4$, it holds that $N = 8$ so that $2^N = 256$ and $Q = 4096$. This is small enough to solve (6.14) *up to a very high precision using `lsqr` in MATLAB. For larger values of M it is no longer tractable to solve this least squares problem.*

Sampling-based evaluation Let $M = 2$ and $u = (1, 0)$. For a given approximate solution $\Psi_{\mathsf{ALS}}^{3,u}$, we want to evaluate $A_{2,\ell}^u(\Psi_{\mathsf{ALS}}^{3,u})$. A single evaluation of (6.13) with Ψ replaced by $\Psi_{\mathsf{ALS}}^{3,u}$ requires on average around $5.1 \cdot 10^{-4}$ seconds. Computing $A_{2,\ell}^u(\Psi_{\mathsf{ALS}}^{3,u})$ for $0 \leq \ell \leq N$ directly would require $2^{24} \approx 1.6 \cdot 10^7$ evaluations of (6.13), i.e. around 2.4 hours. In Section 6.3.4, we proposed two Monte Carlo algorithms to obtain approximations of $A_{2,\ell}^u(\Psi_{\mathsf{ALS}}^{3,u})$. We visualize the variance in the approximation obtained by these algorithms in Figure 6.3. The studied quantity $\frac{2}{N+1} \sum_{\ell=0}^{N} u^T D_s(\frac{\ell}{N}) u$ approximates $\int_0^1 \operatorname{Tr}(D_s(\overline{\rho})) d\overline{\rho}$, where Tr denotes the trace operator. Algorithm 18 clearly leads to a variance reduction. With only 10^5 samples, which can be evaluated in about one minute, we can already reach a variance of 10^{-6}.

6.4.2 Estimation of long-time mean square deviation

In the following, we study estimating the long-term limit using Algorithm 14.

Figure 6.4 motivates our choice of $T = 300$ and $\hat{N}_s = 30\,000$ for the following numerical experiments. The error caused by stopping with $T = 300$ is negligible compared to the stochastic variance when stopping with $\hat{N}_s = 30\,000$.

In Figure 6.5 we analyze how the parameter $\hat{N}_s$ affects the approximation of the quantity

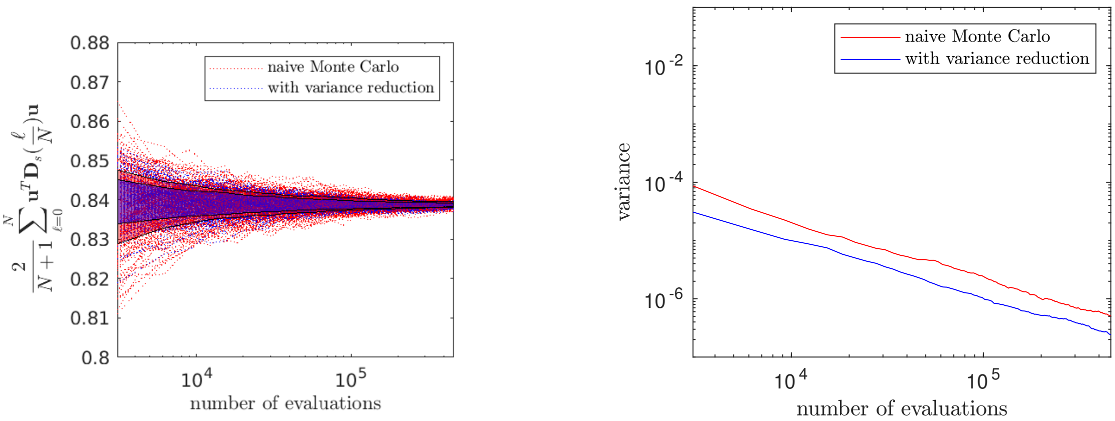


Figure 6.3 – For $M = 2$ and $u = (1, 0)$ we compute $\Psi^{3,u}_{\mathsf{ALS}}$ using Algorithm 16. We then approximate $u^T D_s(\frac{\ell}{N})u = 2A^u_{2,\ell}(\Psi^{3,u}_{\mathsf{ALS}})$ for $0 \le \ell \le N$ using sampling as in Section 6.3.4. We sample $\eta \in C_{M,\ell}$ using two different approaches; from the uniform distribution (naive Monte Carlo) and using Algorithm 18 (with variance reduction). We use a total of 460 800 evaluations of (6.13) for both sampling methods. After every 3 072 evaluations, we evaluate $\frac{2}{N+1}\sum_{\ell=0}^{N} u^T D_s(\frac{\ell}{N})u$ based on the already computed samples. The whole experiment is repeated 250 times. Left: Evolution of $\frac{2}{N+1}\sum_{\ell=0}^{N} u^T D_s(\frac{\ell}{N})u$ with increasing number of samples. The shaded areas mark one standard deviation from the mean for the respective algorithm. Right: Evolution of the variance of $\frac{2}{N+1}\sum_{\ell=0}^{N} u^T D_s(\frac{\ell}{N})u$ with increasing number of samples.

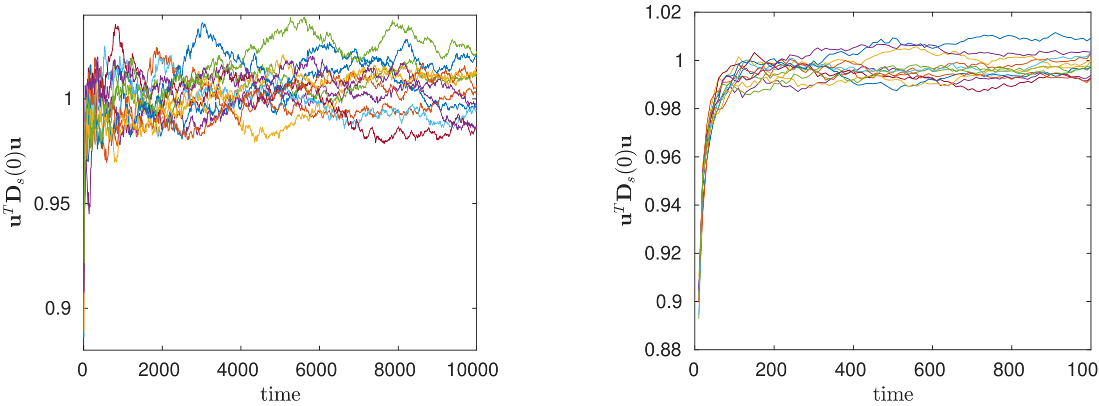


Figure 6.4 – We run Algorithm 14 with $M = 2$, $u = (1, 0)$, $\ell = 0$ and different values for T and $\hat{N}_s$. For each setting, we plot how the approximation of the value $u^T D_s(0)u$, which is known to be equal to 1, evolves over time for 12 random initializations visualized in different colors. Left: $\hat{N}_s = 10\,000$, $T = 10\,0000$. Right: $\hat{N}_s = 100\,000$, $T = 1\,000$.

$\frac{2}{N+1}\sum_{\ell=0}^{N} u^T D_s(\frac{\ell}{N})u$. We observe that the quantity of interest converges to the same value around 0.84 as in Figure 6.3. Note that a single run with $\hat{N}_s = 30\,000$ requires 33 minutes and reaches a variance of 10^{-4}. Compared to the results in Figure 6.3 the computation time needed to achieve the same variance with Algorithm 14 compared to Algorithm 18 is much higher. This is visualized in Figure 6.6.

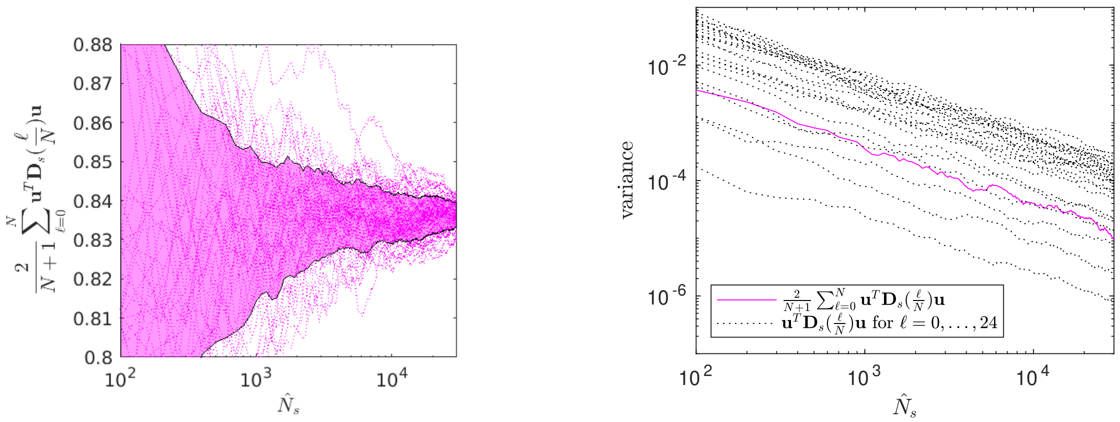


Figure 6.5 – We run Algorithm 14 with $T = 300$, $u = (1, 0)$ for $\ell = 0, \dots, 24$ to approximate $D_s(\frac{\ell}{N})$. We use $\hat{N}_s = 30\,000$ and store intermediate values for $\hat{N}_s = 100, 200, 300, \dots$. Based on these approximations we compute $\frac{2}{N+1}\sum_{\ell=0}^{N} u^T D_s(\frac{\ell}{N})u$. This procedure is repeated 50 times with different random initializations. Left: Evolution of $\frac{2}{N+1}\sum_{\ell=0}^{N} u^T D_s(\frac{\ell}{N})u$ with increasing values of $\hat{N}_s$. The shaded area marks one standard deviation from the mean. Right: Evolution of the variance of $\frac{2}{N+1}\sum_{\ell=0}^{N} u^T D_s(\frac{\ell}{N})u$ with increasing values of $\hat{N}_s$. We additionally display the evolution of the variance of $u^T D_s(\frac{\ell}{N})u$ for individual values of ℓ.

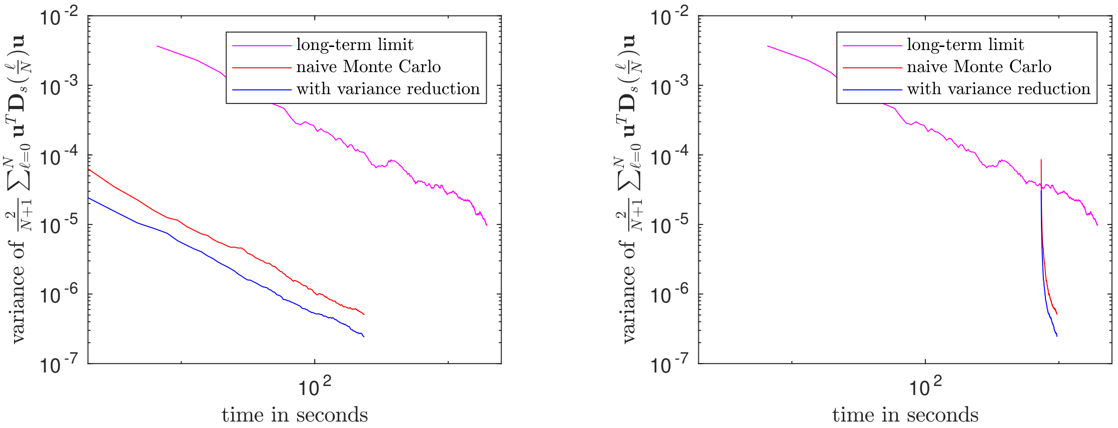


Figure 6.6 – We plot the variances displayed on the right of Figure 6.3 and Figure 6.5. Instead of the number of samples, we display the computation time for evaluating the corresponding number of samples on the x-axis. Left: Computation times only for Algorithm 14, the naive Monte Carlo method and Algorithm 18. Right: We additionally include the computation time of 736 seconds for computing $\Psi_{\mathsf{ALS}}^{3,u}$ using Algorithm 16.

6.4.3 Comparison of algorithms

We now compare the two approaches of approximating the self-diffusion coefficient. We evaluate $u^T D_s(\ell/N)u$ for $u \in \{(1, 0), (0, 1), (1, 1)\}$ to recover all entries of the symmetric matrix $D_s(\ell/N)$. Interpolation of the entries yields an approximation of $D_s(\rho)$ for $\rho \in [0, 1]$. Alternatively, interpolation in the space of symmetric positive definite matrices could be used [232].

For the following experiments, we use Algorithm 14 with a slight modification to approximate $u^T D_s(\overline{\rho})u$ for different values of u simultaneously. This is achieved by keeping track of different α for different u in lines 16 and 17. The optimization problem (6.5) needs to be solved for every u separately, but the solution can be evaluated for different ℓ. In contrast, the sampling in Algorithm 14 needs to be recomputed from scratch for every ℓ.

In Figure 6.7, we display the trace of $D_s(\rho)$ computed using different approaches. Computing the whole matrix $D_s(\ell/N)$ for all $0 \leq \ell \leq N$ took 3 minutes (37 minutes) for the approach based on solving the minimization problem, whereas using Algorithm 14 took 15 minutes (45 minutes) for $M = 1$ ($M = 2$). The much higher variance of Algorithm 14 for approximating the long-term limit leads to clearly visible changes in the graph of trace at the values $\rho = \frac{\ell}{N}$. The figure also contains results using a 6×6 grid ($N = 35$) for which the minimization based approach took 5 hours and 40 and the sampling of the long-term limit took 9 hours and 24 minutes. The sampled solution of the minimization problem changes less due to a smaller sampling variance in Algorithm 18. The variance for both approaches is depicted in Figure 6.8 and listed in Table 6.1. We want to emphasize that the approach based on solving the minimization problem is faster and simultaneously yields a lower variance in the entries of the self-diffusion compared to the estimation of the long-term limit.

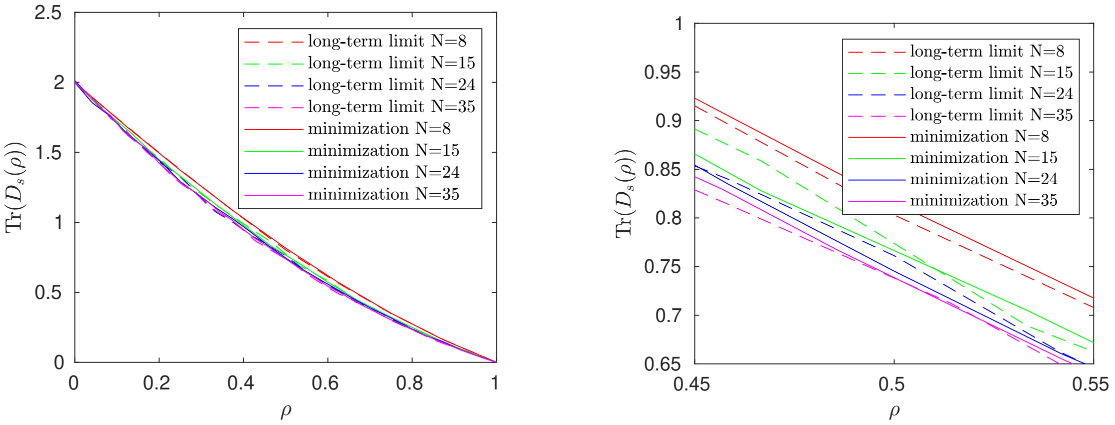


Figure 6.7 – For $M = 1$ ($N = 8$) and $M = 2$ ($N = 24$) we solve the minimization problem (6.5) using Algorithm 16 with $r = 3$. We evaluate $A^u_{M,\ell}$ directly for $M = 1$, whereas for $M = 2$ we use Algorithm 18 with $\widetilde{N}_s = 50$. We compare to Algorithm 14 with $\hat{N}_s = 30\,000$ and $T = 300$ for $M = 1$ and $M = 2$. Repeating this for $u \in \{(1,0),(1,1),(0,1)\}$ yields $D_s(\ell/N)$. Element-wise linear interpolation allows us to evaluate $D_s(\rho)$ for $\rho \in [0,1]$. We plot the trace of $D_s(\rho)$. In addition, we run both algorithms on a 4×4 and 6×6 periodic grid ($N = 15$ and $N = 35$). For $N = 15$ we use the same parameters as for $N = 24$. For $N = 35$ we use rank 10, $\widetilde{N}_s = 150$ and $\hat{N}_s = 80\,000$, $T = 600$. Note that the 4×4 grid lies in between the 3×3 grid for $M = 1$ and the 5×5 grid for $M = 2$. Right: Zoom on part of the graphs.

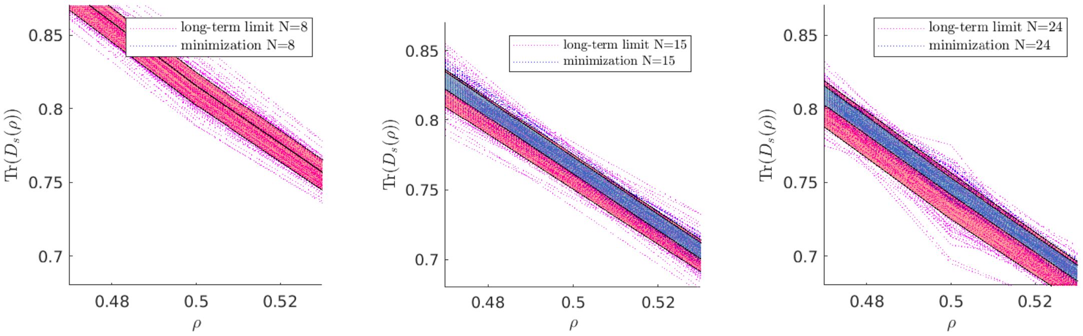


Figure 6.8 – We repeat the experiment presented in Figure 6.7 100 times for $N = 8$, $N = 15$ and $N = 24$. We plot a zoom on a part of the graphs. The shaded areas mark one standard deviation from the mean for each of the methods.

		$\max\limits_{\ell\in\{0,\dots,N\}} \mathrm{Var}(\mathrm{Tr}(D_s(\ell/N)))$	$\frac{1}{N+1}\sum_{\ell=0}^{N} \mathrm{Var}(\mathrm{Tr}(D_s(\ell/N)))$
N = 8	long-term limit	$2.313 \cdot 10^{-4}$	$1.072 \cdot 10^{-4}$
	minimization	$8.791 \cdot 10^{-9}$	$4.231 \cdot 10^{-9}$
N = 15	long-term limit	$3.433 \cdot 10^{-4}$	$1.410 \cdot 10^{-4}$
	minimization	$2.548 \cdot 10^{-4}$	$6.835 \cdot 10^{-5}$
N=24	long-term limit	$4.377 \cdot 10^{-4}$	$1.989 \cdot 10^{-4}$
	minimization	$3.262 \cdot 10^{-4}$	$8.001 \cdot 10^{-5}$

Table 6.1 – We display the mean and maximum of the variance for the 100 approximations of the self-diffusion coefficient computed in Figure 6.8. Note that we do not need to sample to evaluate $A^u_{M,\ell}$ for $N = 8$. The variance for minimization with $N = 8$ is solely caused by the random initializations of R^0_s in Algorithm 17.

6.5 Application: Cross-diffusion system

In this section, we recall how self-diffusion matrices arise in the context of cross-diffusion systems. We then use our novel low-rank approach described in Section 6.3 to approximate the self-diffusion matrices needed to resolve a slightly simplified cross-diffusion system with a cell-centered finite volume method.

6.5.1 Hydrodynamic limit of a lattice-based stochastic hopping model

Lattice-based stochastic hopping models [214, 266, 301] describe the evolution of a mixture of multi-species particles, the positions of which are clamped on a (periodic) lattice $T_n := \{0, \dots, \frac{n-1}{n}\}^d$, $d = 1, 2, 3$. After a certain amount of time, a single particle jumps to a nearby node according to a given Markovian random walk. Let $v_k := ((v_k)_i)_{1\leq i\leq d} \in \mathbb{Z}^d\backslash\{0\}$ for $k = 1, \dots, K$ be the displacement vectors for all possible jumps that one particle can

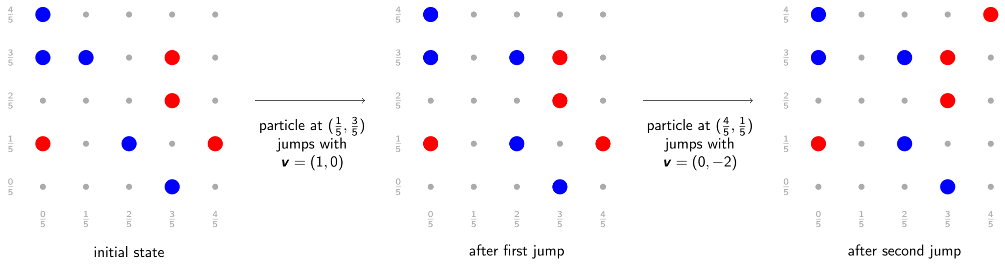

$\frac{4}{5}$
$\frac{3}{5}$
$\frac{2}{5}$
$\frac{1}{5}$
$\frac{0}{5}$
$\frac{0}{5}$ $\frac{1}{5}$ $\frac{2}{5}$ $\frac{3}{5}$ $\frac{4}{5}$
initial state
particle at $(\frac{1}{5}, \frac{3}{5})$ jumps with $\mathbf{v} = (1, 0)$
after first jump
particle at $(\frac{4}{5}, \frac{1}{5})$ jumps with $\mathbf{v} = (0, -2)$
after second jump

Figure 6.9 – The evolution of state of the particles on a lattice with $n = 5, d = 2$. Left: Initial state. Middle: After the blue particle at $s = (\frac{1}{5}, \frac{3}{5})$ jumped with $\mathbf{v} = (1, 0)$ to $s + \frac{1}{5}\mathbf{v}$. Right: After the red particle at position $(\frac{4}{5}, \frac{1}{5})$ jumped with $\mathbf{v} = (0, -2)$. Since the lattice is periodic the particle ends up in position $(\frac{4}{5}, \frac{4}{5})$.

make, i.e. a particle at position $x \in T_n$ lands at position $x + \frac{1}{n}v_k \in T_n$. The rate of jumping in the direction $\frac{1}{n}v_k$ is given by $p_k \in]0, 1]$. We assume that the particles can be of two possible species, either species `blue` or species `red`. We study how the distribution of the particles belonging to each species evolves over time in the limit when the number of lattice points and particles tends to infinity. See Figure 6.9 for a schematic illustration of this lattice-hopping process.

We are interested in simulating the hydrodynamic limit of this lattice-based hopping model, which was identified in [266]. In the limit $n \to +\infty$, this hydrodynamic limit reads as a cross-diffusion system defined on the d-dimensional torus $T^d := (\mathbb{R}/\mathbb{Z})^d$. At this scale, we consider densities $\rho_{\texttt{red}} : T^d \times \mathbb{R}_+ \to [0, 1]$ and $\rho_{\texttt{blue}} : T^d \times \mathbb{R}_+ \to [0, 1]$, where $\rho_{\texttt{red}}(x, t)$ (respectively $\rho_{\texttt{blue}}(x, t)$) denotes the local volumic fraction of `red` (respectively `blue`) particles at point $x \in T^d$ and time $t \geq 0$. The local volumic fractions $\rho_{\texttt{red}}$ and $\rho_{\texttt{blue}}$ are shown to be solutions to the following cross-diffusion system [266]

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho_{\texttt{red}} \\ \rho_{\texttt{blue}}\end{pmatrix} = \frac{1}{2}\nabla \cdot \left[\begin{pmatrix} \frac{\rho_{\texttt{blue}}}{\rho}D_s(\rho) + \frac{\rho_{\texttt{red}}}{\rho}D & \frac{\rho_{\texttt{red}}}{\rho}(D - D_s(\rho)) \\ \frac{\rho_{\texttt{blue}}}{\rho}(D - D_s(\rho)) & \frac{\rho_{\texttt{red}}}{\rho}D_s(\rho) + \frac{\rho_{\texttt{blue}}}{\rho}D \end{pmatrix}\begin{pmatrix}\nabla\rho_{\texttt{red}} \\ \nabla\rho_{\texttt{blue}}\end{pmatrix}\right], \tag{6.15}$$

in $T^d \times \mathbb{R}_+$, where $\rho := \rho_{\texttt{red}} + \rho_{\texttt{blue}}$. The matrix $D \in \mathbb{R}^{d\times d}$ is defined such that $D_{ij} := \sum_{k=1}^K p_k(v_k)_i(v_k)_j$ for all $i, j = 1, \dots, d$. The matrix $D_s(\rho)$ denotes the self-diffusion matrix application (6.1) of the tagged particle process with jump directions v_k and jump probabilities p_k for $k = 1, \dots, K$.

We complement system (6.15) by the initial conditions $\rho_{\texttt{red}}(0, x) = \rho^0_{\texttt{red}}(x)$, $\rho_{\texttt{blue}}(0, x) = \rho^0_{\texttt{blue}}(x)$, where $\rho^0_{\texttt{red}}, \rho^0_{\texttt{blue}} \in L^\infty(T^d)$ satisfy $0 \leq \rho^0_{\texttt{red}} + \rho^0_{\texttt{blue}} \leq 1$ almost everywhere in T^d. In this case, it can be shown [266] that there exists a unique solution $(\rho_{\texttt{red}}, \rho_{\texttt{blue}}) \in L^\infty(T^d \times \mathbb{R}_+)^2$ to system (6.15). In addition, it holds that for almost all $t \geq 0$ and $x \in T^d$, $0 \leq \rho_{\texttt{red}}(x, t)$, $0 \leq \rho_{\texttt{blue}}(x, t)$ and $\rho_{\texttt{red}}(x, t) + \rho_{\texttt{blue}}(x, t) \leq 1$.

6.5.2 Deterministic resolution of a simplified cross-diffusion system

Since, the design and numerical analysis of a numerical scheme for the resolution of the cross-diffusion system (6.15) is still subject to future research, we present a numerical scheme to resolve a simplified system. In this section, we develop a cell-centred finite volume method [115, 116] to simulate this simplified cross-diffusion system, assuming that a numerical approximation of the self-diffusion matrix $D_s(\overline{\rho})$ can be computed for any $\overline{\rho} \in [0,1]$.

Simplified cross-diffusion system

Let $\Omega \subset \mathbb{R}^d$ be a polyhedric bounded domain of $\mathbb{R}^d$. The local volumic fractions of `red` and `blue` particles are now given by functions $\rho_{\texttt{red}} : \Omega \times \mathbb{R}_+ \to [0,1]$ and $\rho_{\texttt{blue}} : \Omega \times \mathbb{R}_+ \to [0,1]$. These functions are assumed to satisfy the following simplified cross-diffusion system:

$$\begin{aligned}
&\frac{\partial}{\partial t}\begin{pmatrix}\rho_{\texttt{red}} \\ \rho_{\texttt{blue}}\end{pmatrix} = \\
&\quad \frac{1}{2}\nabla \cdot \left[\begin{pmatrix} \mathrm{Tr}\left[\frac{\rho_{\texttt{blue}}}{\rho} D_s(\rho) + \frac{\rho_{\texttt{red}}}{\rho} D\right] & \mathrm{Tr}\left[\frac{\rho_{\texttt{red}}}{\rho}(D - D_s(\rho))\right] \\ \mathrm{Tr}\left[\frac{\rho_{\texttt{blue}}}{\rho}(D - D_s(\rho))\right] & \mathrm{Tr}\left[\frac{\rho_{\texttt{red}}}{\rho} D_s(\rho) + \frac{\rho_{\texttt{blue}}}{\rho} D\right] \end{pmatrix}\begin{pmatrix}\nabla\rho_{\texttt{red}} \\ \nabla\rho_{\texttt{blue}}\end{pmatrix}\right],
\end{aligned} \tag{6.16}$$

in $\Omega \times \mathbb{R}_+$, where $\rho := \rho_{\texttt{red}} + \rho_{\texttt{blue}}$. We complement system (6.15) by the initial conditions $\rho_{\texttt{red}}(0,x) = \rho^0_{\texttt{red}}(x)$, $\rho_{\texttt{blue}}(0,x) = \rho^0_{\texttt{blue}}(x)$, where $\rho^0_{\texttt{red}}, \rho^0_{\texttt{blue}} \in L^\infty(\Omega)$ satisfying $0 \le \rho^0_{\texttt{red}} + \rho^0_{\texttt{blue}} \le 1$ almost everywhere in Ω. We also assume that system (6.16) is complemented with Neumann boundary conditions instead of periodic boundary conditions.

Time discretization

Let $T_f > 0$ denote some final time. For the time discretization, we introduce a division of the interval $[0, T_f]$ into subintervals $I_p := [t_{p-1}, t_p]$, $1 \le p \le P_t$ for some $P_t \in \mathbb{N}$ such that $0 = t_0 < t_1 < \cdots < t_{P_t} = T_f$. The time steps are denoted by $\Delta t_p = t_p - t_{p-1}$, $p = 1, \dots, P_t$. For a given sequence of real numbers $(v^p)_{p\in\mathbb{N}}$, we define the approximation of the first-order time derivative thanks to the backward Euler scheme as follows:

$$\partial_t v^p := \frac{v^p - v^{p-1}}{\Delta t_p} \quad \forall\, 1 \le p \le P_t.$$

Space discretization

For the space discretization, we consider a conforming simplicial mesh $\mathcal{T}_h$ of the domain Ω, i.e. $\mathcal{T}_h$ is a set of elements K verifying $\displaystyle\bigcup_{K\in\mathcal{T}_h} \overline{K} = \overline{\Omega}$, where the intersection of the closure of two elements of $\mathcal{T}_h$ is either an empty set, a vertex, or an l-dimensional face, $0 \le l \le d-1$. Denote by h_K the diameter of the generic element $K \in \mathcal{T}_h$ and $h = \max_{K\in\mathcal{T}_h} h_K$. We denote by $\mathcal{E}_h$ the set of mesh faces. Boundary faces are collected in the set $\mathcal{E}_h^{\mathrm{ext}} = \{\sigma \in \mathcal{E}_h; \sigma \subset \partial\Omega\}$ and internal faces are collected in the set $\mathcal{E}_h^{\mathrm{int}} = \mathcal{E}_h \backslash \mathcal{E}_h^{\mathrm{ext}}$. To each face $\sigma \in \mathcal{E}_h$, we associate a unit normal vector n_σ; for $\sigma \in \mathcal{E}_h^{\mathrm{int}}$, $\sigma = K \cap L$, n_σ points from K towards L and for $\sigma \in \mathcal{E}_h^{\mathrm{ext}}$ it coincides with the outward unit normal vector n_Ω of Ω. We also denote by N_{e} the number of elements in the mesh $\mathcal{T}_h$. Furthermore, the notation $n_{K,\sigma}$ stands for the outward unit normal vector to the element K on σ. We also assume that the family $\mathcal{T}_h$ is superadmissible in the sense that for all cells $K \in \mathcal{T}_h$ there exists a point $x_K \in K$ (the cell center) and for all edges $\sigma \in \mathcal{E}_h$ there exists a point $x_\sigma \in \partial K$ (the edge center) such that, for all edges $\sigma \in \mathcal{E}_K$, the line segment joining x_K with x_σ is orthogonal to σ (see [116]). For an interior edge $\sigma \in \mathcal{E}_h^{\mathrm{int}}$ shared by two elements K and L (denoted in the sequel by $\sigma = K \cap L$), we define the distance between these elements $d_{KL} := \mathrm{dist}(x_K, x_L)$. Figure 6.10 provides a schematic illustration.

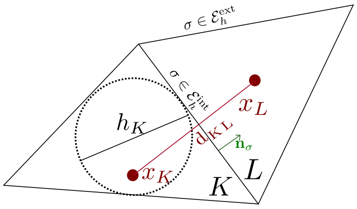

$\sigma \in \mathcal{E}_{\mathrm{h}}^{\mathrm{ext}}$
$\sigma \in \mathcal{E}_{\mathrm{h}}^{\mathrm{int}}$
x_L
h_K
d_{KL}
$\mathbf{n}_\sigma$
x_K
L
K

Figure 6.10 – Illustration of the notation for a mesh with two elements.

The cell-centered finite volume method

In the context of the cell-centered finite volume method, the unknowns of the model are discretized using one value per cell: $\forall 1 \le p \le P_t$ we let

$$U^p := (U_K^p)_{K\in\mathcal{T}_h} \in \mathbb{R}^{2N_{\mathrm{e}}}, \quad \text{with} \quad U_K^p := (\rho_{\texttt{red},K}^p, \rho_{\texttt{blue},K}^p) \in \mathbb{R}^2$$

where $\rho_{\texttt{red},K}^p$ and $\rho_{\texttt{blue},K}^p$ are respectively the discrete elementwise unknowns approximating the values of $\rho_{\texttt{red}}(t_p)$ and $\rho_{\texttt{blue}}(t_p)$ in the element $K \in \mathcal{T}_h$. More precisely, $\rho_{\texttt{red},K}^p \approx \dfrac{1}{|K|}\displaystyle\int_K \rho_{\texttt{red}}(t_p, x)\,\mathrm{d}x$ and $\rho_{\texttt{blue},K}^p \approx \dfrac{1}{|K|}\displaystyle\int_K \rho_{\texttt{blue}}(t_p, x)\,\mathrm{d}x$. Integrating (6.15) over the element $K \in \mathcal{T}_h$, using next the Green formula and the Neumann boundary conditions, the cell-centered finite volume scheme we solve is the following: for a given fixed value of U^{p-1}, find $U^p \in \mathbb{R}^{2N_e}$ such that

$$\begin{aligned} H_{1,K}(U^p) &= |K|\partial_t \rho^p_{\texttt{red},K} - \frac{1}{2}\sum_{\sigma\in\mathcal{E}_K} \mathcal{F}_{1,K,\sigma}(U^p) = 0 \quad \forall K \in \mathcal{T}_h,\\ H_{2,K}(U^p) &= |K|\partial_t \rho^p_{\texttt{blue},K} - \frac{1}{2}\sum_{\sigma\in\mathcal{E}_K} \mathcal{F}_{2,K,\sigma}(U^p) = 0 \quad \forall K \in \mathcal{T}_h. \end{aligned} \tag{6.17}$$

The numerical fluxes $\mathcal{F}_{1,K,\sigma}(U^p)$ and $\mathcal{F}_{2,K,\sigma}(U^p)$ are respectively approximations of

$$\begin{aligned} \mathcal{F}_{1,K,\sigma}(U^p) &\approx \left(S^{11}(U^p)\boldsymbol{\nabla}\rho^p_{\texttt{red}} + S^{12}(U^p)\boldsymbol{\nabla}\rho^p_{\texttt{blue}}\right)\cdot n_{K,\sigma}\\ \mathcal{F}_{2,K,\sigma}(U^p) &\approx \left(S^{21}(U^p)\boldsymbol{\nabla}\rho^p_{\texttt{red}} + S^{22}(U^p)\boldsymbol{\nabla}\rho^p_{\texttt{blue}}\right)\cdot n_{K,\sigma}, \end{aligned}$$

where $S^{11}(U^p) := \left(\mathcal{S}^{11}_K(U^p)\right)_{K\in\mathcal{T}_h}$, $S^{12}(U^p) := \left(\mathcal{S}^{12}_K(U^p)\right)_{K\in\mathcal{T}_h}$, $S^{21}(U^p) := \left(\mathcal{S}^{21}_K(U^p)\right)_{K\in\mathcal{T}_h}$, $S^{22}(U^p) := \left(\mathcal{S}^{22}_K(U^p)\right)_{K\in\mathcal{T}_h}$ are collections of real numbers defined on each cell of the mesh as follows. For all $K \in \mathcal{T}_h$,

$$\begin{aligned} \mathcal{S}^{11}_K(U^p) &:= \mathrm{Tr}\left[\frac{\rho^p_{\texttt{blue},K}}{\rho^p_K} D_s(\rho^p_K) + \frac{\rho^p_{\texttt{red},K}}{\rho^p_K} D\right],\\ \mathcal{S}^{12}_K(U^p) &:= \mathrm{Tr}\left[\frac{\rho^p_{\texttt{red},K}}{\rho^p_K}\left(D - D_s(\rho^p_K)\right)\right]\\ \mathcal{S}^{21}_K(U^p) &:= \mathrm{Tr}\left[\frac{\rho^p_{\texttt{blue},K}}{\rho^p_K}\left(D - D_s(\rho^p_K)\right)\right]\\ \mathcal{S}^{22}_K(U^p) &:= \mathrm{Tr}\left[\frac{\rho^p_{\texttt{red},K}}{\rho^p_K} D_s(\rho^p_K) + \frac{\rho^p_{\texttt{blue},K}}{\rho^p_K} D\right], \end{aligned}$$

where $\rho^p_K := \rho^p_{\texttt{red},K} + \rho^p_{\texttt{blue},K}$. Now we are interested in computing an approximation of $\left(S^{ij}(U^p)\boldsymbol{\nabla}\rho^p_\sharp\right)\cdot n_{K,\sigma}$ for all $1 \le i,j \le 2$ and $\sharp \in \{\texttt{red}, \texttt{blue}\}$.

For any edge $\sigma \in \mathcal{E}^{\mathrm{int}}_h$, denoting by $K, L \in \mathcal{T}_h$ the two cells sharing the edge σ, we employ the following harmonic averaging formula for all $1 \le i,j \le 2$,

$$\left(S^{ij}(U^p)\boldsymbol{\nabla}\rho^p_\sharp\right)\cdot n_{K,\sigma} \approx |\sigma|\frac{\mathcal{S}^{ij}_K(U^p)\mathcal{S}^{ij}_L(U^p)}{d_{K,\sigma}\mathcal{S}^{ij}_L(U^p) + d_{L,\sigma}\mathcal{S}^{ij}_K(U^p)}\left(\rho^p_{\sharp,L} - \rho^p_{\sharp,K}\right).$$

We refer to [1, 113, 114] for more details. This averaging formula enables then to obtain the expression of the fluxes $\mathcal{F}_{1,K,\sigma}(U^p)$ and $\mathcal{F}_{2,K,\sigma}(U^p)$.

At each time step p, we thus have to solve the nonlinear problem: find $U^p \in \mathbb{R}^{2N_e}$ such that $\mathcal{H}^p(U^p) = 0$, where $\mathcal{H}^p(U^p) := (H_{1,K}(U^p), H_{2,K}(U^p))_{K\in\mathcal{T}_h} \in \mathbb{R}^{2N_e}$, where $H_{1,K}(U^p)$ and $H_{2,K}(U^p)$ are defined by (6.17).

Resolution of the nonlinear problem by a Newton method

In this section, we present a Newton procedure for computing the approximate solution U^p at each time step p. For $1 \leq p \leq P_t$ and $U^{p,0} \in \mathbb{R}^{2N_\mathrm{e}}$ fixed (typically $U^{p,0} = U^{p-1}$), the Newton algorithm generates a sequence $(U^{p,k})_{k\geq 1}$, with $U^{p,k} \in \mathbb{R}^{2N_\mathrm{e}}$ given by the system of linear algebraic equations:

$$A^{p,k-1}U^{p,k} = B^{p,k-1}. \tag{6.18}$$

Here, the Jacobian matrix $A^{p,k-1} \in \mathbb{R}^{2N_\mathrm{e},2N_\mathrm{e}}$ and the right-hand side vector $B^{p,k-1} \in \mathbb{R}^{2N_\mathrm{e}}$ are defined by

$$A^{p,k-1} := J_{\mathcal{H}^p}(U^{p,k-1}) \quad \text{and} \quad B^{p,k-1} := J_{\mathcal{H}^p}(U^{p,k-1})U^{p,k-1} - \mathcal{H}^p(U^{p,k-1}).$$

Note that $J_{\mathcal{H}^p}(U^{p,k-1})$ is the Jacobian matrix of the function $\mathcal{H}^p$ at point $U^{p,k-1}$. A classical stopping criterion for system (6.18) is for instance

$$\left\|\mathcal{H}^p(U^{p,k})\right\|_2 / \left\|\mathcal{H}^p(U^{p,0})\right\|_2 < \varepsilon_{\mathrm{lin}} \tag{6.19}$$

where $\varepsilon_{\mathrm{lin}} > 0$ is small enough. We refer to the book [185] for large descriptions on linearization techniques.

6.5.3 Numerical Experiment

In the following numerical experiment[†], we solve the PDE system (6.16) using the finite volume scheme described in this section. A approximation of the self-diffusion coefficient $D_s(\overline{\rho})$ is obtained from $\Psi^{1,u}_{\mathsf{ALS}}$ as in Section 6.4. We set the domain $\Omega := [0,1] \times [0,1]$ and consider a uniform spatial mesh ($N_\mathrm{e} = 312$ elements). We use a constant time step $\Delta t_p = \Delta t = 10^{-3}$ seconds $\forall 1 \leq p \leq P_t$. The final time of simulation is $T_f = 10$ seconds. We employ at each time step $1 \leq p \leq P_t$ a Newton solver as described in Section 6.5.2 with $\varepsilon_{\mathrm{lin}} = 10^{-8}$. The initial values are defined by

$$\rho^0_{\texttt{red}}(x,y) := 0.25+0.25\cos(\pi x)\cos(\pi y) \quad \text{and} \quad \rho^0_{\texttt{blue}}(x,y) := 0.5-0.5\cos(\pi x)\cos(\pi y).$$

Figure 6.11 displays the shape of the numerical solutions $\rho_{\texttt{red}}$ and $\rho_{\texttt{blue}}$ for several time steps. We observe that the local volumic fractions evolve over time to reach constant profiles (around 0.25 for $\rho_{\texttt{red}}$ and 0.5 for $\rho_{\texttt{blue}}$) in the long time limit. This indicates, that our scheme is stable in the sense that the densities lie in their physical ranges. In every time step, only 2 or 3 Newton iterations are required to reach the stopping criterion (6.19).

[†] The code to reproduce these results is available from https://github.com/cstroessner/SelfDiffusionCoefficent.git

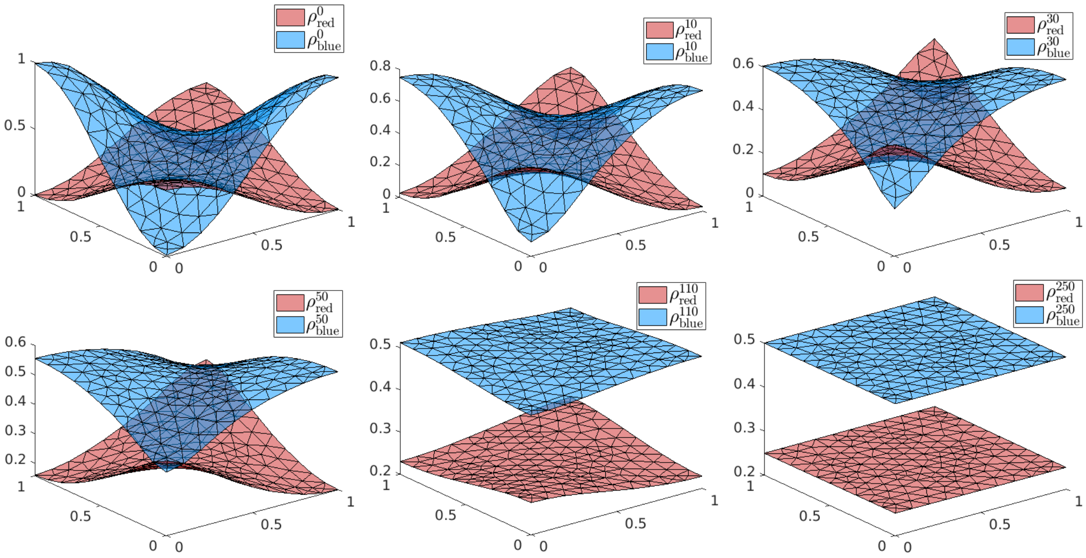


Figure 6.11 – Solutions $\rho_{\texttt{red}}$ and $\rho_{\texttt{blue}}$ for several time steps. Top left $p = 0$, top middle $p = 10$, top right $p = 30$, bottom left $p = 50$, bottom middle $p = 110$, bottom right $p = 250$.

7 Fast global spectral method

Global spectral methods [122, 143, 319, 326] offer the potential to solve PDEs to very high accuracy. These methods are based on studying the impact of the differential operator when applied to a finite set of basis functions such as tensorized Chebyshev polynomials with bounded degree (2.16). We then compute a linear combination of these basis functions that (approximately) satisfies the PDE. For one- and two-dimensional rectangular domains this approach led to the development of the solvers Chebop [102, 103, 246] and Chebop2 [319] contained in the Chebfun package [104]. Extensions to triangular domains and to disks have been derived in [244, 245, 338]. In the three-dimensional setting only specialised solvers for the Poissons's equation have been developed so far [125, 321, 346].

This, chapter is concerned with the numerical solution of general linear PDEs on cubes of the form

$$\mathcal{L}u = f \quad \text{on } [-1,1]^3, \tag{7.1}$$

complemented with linear boundary conditions. Note that even though the domains of three-dimensional PDEs arising in medicine [166, 219, 311], engineering [28, 107, 263] and geosciences [175, 226, 347] are rarely cubes, they can often be mapped onto cubes [124]. Following the idea of global spectral methods, we approximate both the solution u and the right-hand side f by multivariate polynomials of the form (1.3). We approximate the operator $\mathcal{L}$ by a mapping mimicking the impact of applying $\mathcal{L}$ on the level of the coefficient tensors. In order to define this mapping, we express the three-dimensional differential operator as a combination of one-dimensional linear differential operators, which we determine using a CP decomposition (2.4). By representing u in a Chebyshev basis (2.16) and f in an ultraspherical basis, we can express the action of each of these one-dimensional operators on the coefficients in terms of sparse and well-conditioned differentiation and multiplication matrices [246]. The combination of applying these matrices yields the desired mapping. Solving the PDE corresponds to inverting this mapping, which can be seen as solving a tensor-valued linear system. Discretizing the boundary conditions provides additional linear constraints under which inverting the linear system has a unique solution. Using substitution we obtain an unconstrained tensor-valued linear system uniquely determining a subtensor, from which we recover the full coefficient tensor of the solution. We would like to point out that our discretization approach can be applied to a wide variety of PDEs, but it does not necessarily preserve certain desirable properties of the differential operator.

It turns out that our discretization of the PDE (7.1) has a Kronecker structure that can often be exploited. For instance, Poisson's equation with homogeneous Dirichlet boundary conditions leads to a Laplace-like equation [24, 70, 206, 236]. Laplace-like equations can be solved efficiently using the blocked recursive algorithm in [69]. This algorithm is asymptotically slower than the nested alternating direction implicit method in [125], but our numerical experiments in Section 7.4 demonstrate that the recursive blocked algorithm is significantly faster in practice. Even if the PDE of interest does not immediately lead to a structure suitable for the fast solver, we can often apply the blocked recursive algorithm as a preconditioner for GMRES.

The algorithmic ideas presented in this chapter, can be extended from solving stationary linear PDEs, such as the Helmholtz equation and (convection) diffusion problems, to solving time-dependent PDEs and PDE eigenvalue problems of the form

$$\frac{\partial}{\partial t}u = \mathcal{L}u \quad \text{and} \quad \mathcal{L}u = \lambda u,$$

which we solve using implicit Euler and inverse iteration methods, respectively. We expect to obtain accurate approximations of the solution, when both the PDE coefficients and the solution are sufficiently smooth to be well approximated by truncated expansions with tensorized polynomial basis functions. For non-smooth solutions, our global spectral method might lead to inaccurate approximations.

Remark 7.1. *Our global spectral method can be used to compute solutions numerically to very high accuracy. It is not to be confused with so-called spectral element methods [124, 164, 184] and p- and hp-finite element methods [18, 244, 342]. In those methods, u is not approximated globally by a truncated series expansion. Instead, u is written as sum of (locally supported) functions, each of which is approximated individually by a truncated series expansion. There also exist solvers relying on using domain decompositions in combination with truncated series expansions [160, 259]. We want to emphasize that the methods presented in this work can be used as local solver, when the elements/subdomains can be mapped to cubes.*

This chapter is based on the article [304]. Its remainder is structured as follows. In Section 7.1, we define the linear differential operator and the approximation format. In Section 7.2, we derive the mapping on the level of the coefficient tensor. The discretization of PDEs and the efficient solution of the resulting tensor-valued linear system is discussed in Section 7.3. In Section 7.4, we apply our global spectral method to solve stationary PDEs, parabolic PDEs and PDE eigenvalue problems numerically to very high accuracy.

7.1 Problem setting

7.1.1 Structure of a linear differential operator

A linear partial differential operator $\mathcal{L}$ on the domain $[-1,1]\times[-1,1]\times[-1,1]$ maps a sufficiently smooth function $u:[-1,1]^3\to\mathbb{R}$ to

$$\mathcal{L}u(x,y,z)=\sum_{a=0}^{N_x}\sum_{b=0}^{N_y}\sum_{c=0}^{N_z}\alpha_{abc}(x,y,z)\frac{\partial^{a+b+c}}{\partial x^a\partial y^b\partial z^c}u(x,y,z), \tag{7.2}$$

where N_x,N_y,N_z are called differential order and $\alpha_{abc}(x,y,z):[-1,1]^3\to\mathbb{R}$ are coefficient functions for $0\le a\le N_x,\ 0\le b\le N_y,\ 0\le c\le N_z$. For the differential operator $\mathcal{L}$ we consider the linear PDE

$$\mathcal{L}u=f \tag{7.3}$$

with right hand side $f:[-1,1]^3\to\mathbb{R}$. The PDE can be solved uniquely when the system is complemented with sufficient boundary conditions.

7.1.2 Approximation format

We approximate the solution $u:[-1,1]^3\to\mathbb{R}$ of the PDE (7.3) in the space $\mathbb{P}_{n_1,n_2,n_3}$ of trivariate polynomials of degree at most (n_1,n_2,n_3). We express u in terms of a tensorized basis of Chebyshev polynomials (see Section 2.3), which leads to a representation of the form

$$u(x,y,z)\approx\sum_{i=0}^{n_1}\sum_{j=0}^{n_2}\sum_{k=0}^{n_3}\mathcal{U}_{ijk}T_i(x)T_j(y)T_k(z), \tag{7.4}$$

with coefficient tensor $\mathcal{U}\in\mathbb{R}^{(n_1+1)\times(n_2+1)\times(n_3+1)}$.

7.2 Operator discretization

In the following, we discretize the differential operator $\mathcal{L}$ for fixed coefficient functions α_{abc} by approximating $\mathcal{L}$ as mapping from $\mathbb{P}_{n_1,n_2,n_3}$ to $\mathbb{P}_{n_1,n_2,n_3}$. This is particularly easy for constant coefficients in the differential operator $\mathcal{L}$, i.e., $\alpha_{abc}(x,y,z)=\alpha_{abc}\in\mathbb{R}$ in Equation (7.2). Applying a linear differential operator with constant coefficients to a polynomial does not increase the polynomial degree. For non-constant coefficients an additional truncation is needed to obtain a polynomial in $\mathbb{P}_{n_1,n_2,n_3}$. Thus, it is natural to see the application of the operator as a transformation of the coefficient tensor. We discuss the discretization for the constant case first before generalizing to the non-constant case in Section 7.2.3.

7.2.1 One-dimensional differential operators

We briefly recapitulate how to obtain the mapping describing the transformation of coefficients in a one-dimensional setting before returning to the three-dimensional setting. Let $u : [-1, 1] \to \mathbb{R}$ be a polynomial of degree n represented in the Chebyshev basis by $u(x) = \sum_{k=0}^{n} u_k T_k(x)$, with coefficients $u_k \in \mathbb{R}$. Applying a one-dimensional linear differential operator $\mathcal{L}$ of order N with constant coefficients to u can be written as

$$\mathcal{L}u(x) = \sum_{a=0}^{N} \alpha_a \frac{d^a}{dx^a} u(x), \tag{7.5}$$

with coefficients $\alpha_a \in \mathbb{R}$. This is a linear combination of (higher order) derivatives of u. For every derivative of the polynomial u, there exists a so-called differentiation matrix which maps the coefficient vector $\mathbf{u} = (u_0, \ldots, u_n)$ to the coefficients of the derivative. The remainder of this section follows the ideas of [246] to represent the derivative using an ultraspherical basis instead of a Chebyshev basis, which leads to better conditioned and sparse differentiation matrices.

For the parameter $\lambda > 0$ and $k = 0, 1, \ldots,$ ultraspherical polynomials follow the recurrence relation

$$\begin{aligned} &4\lambda(k + \lambda + 1)(1 - x^2)C_k^{(\lambda+1)}(x) = \\ &\quad - (k + 1)(k + 2)C_{k+2}^{(\lambda)}(x) + (k + 2\lambda)(k + 2\lambda + 1)C_k^{(\lambda)}(x), \end{aligned}$$

where $C_k^{(1)}(x) = (\sin(k + 1)\cos^{-1}(x))/\sin(\cos^{-1}(x))$ [240]. Let $v(x) = \sum_{k=0}^{n} v_k C_k^{(\lambda)}(x)$ denote the λth derivative of u represented in $C^{(\lambda)}$ basis, then the coefficient vector $\mathbf{v} = (v_0, \ldots, v_n)$ satisfies $\mathbf{v} = D_\lambda \mathbf{u}$, where the sparse differentiation matrix $D_\lambda \in \mathbb{R}^{(n+1)\times(n+1)}$

is defined as

$$D_\lambda = 2^{\lambda-1}(\lambda-1)! \begin{pmatrix} \overbrace{0\dots0}^{\lambda \text{ times}} & \lambda & & & \\ & & \lambda+1 & & \\ & & & \ddots & \\ & & & & n \\ & & & & 0 \\ & & & & \vdots \\ & & & & 0 \end{pmatrix}.$$

Note that the ultraspherical basis is different for different λ. The sparse transformation matrices $S_0 \in \mathbb{R}^{(n+1)\times(n+1)}$, mapping Chebyshev to $C^{(1)}$ coefficients, and $S_\lambda \in \mathbb{R}^{(n+1)\times(n+1)}$, mapping $C^{(\lambda)}$ to $C^{(\lambda+1)}$, are defined as

$$S_0 = \begin{pmatrix} 1 & 0 & -\frac{1}{2} & & & & \\ & \frac{1}{2} & 0 & -\frac{1}{2} & & & \\ & & \frac{1}{2} & 0 & -\frac{1}{2} & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \frac{1}{2} & 0 & -\frac{1}{2} \\ & & & & & \frac{1}{2} & 0 \\ & & & & & & \frac{1}{2} \end{pmatrix},$$

$$S_\lambda = \begin{pmatrix} 1 & 0 & -\frac{\lambda}{\lambda+2} & & & & \\ & \frac{\lambda}{\lambda+1} & 0 & -\frac{\lambda}{\lambda+3} & & & \\ & & \frac{\lambda}{\lambda+2} & 0 & -\frac{\lambda}{\lambda+4} & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \frac{\lambda}{\lambda+n-2} & 0 & -\frac{\lambda}{\lambda+n} \\ & & & & & \frac{\lambda}{\lambda+n-1} & 0 \\ & & & & & & \frac{\lambda}{\lambda+n} \end{pmatrix}.$$

Let $\mathcal{L}u(x) = \sum_{k=0}^{n} w_k C_k^{(N)}(x)$ be represented in $C^{(N)}$ ultraspherical basis with coefficient vector $\mathbf{w} = (w_0, \dots, w_n)$, then

$$\mathbf{w} = (\underbrace{a_N D_N + a_{N-1} S_{N-1} D_{N-1} + \dots + a_1 S_{N-1} \cdots S_1 D_1 + a_0 S_{N-1} \cdots S_0}_{=L})\mathbf{u}. \quad (7.6)$$

The matrix $L \in \mathbb{R}^{(n+1)\times(n+1)}$ describes how the one-dimensional differential operator $\mathcal{L}$ acts on Chebyshev coefficients of the solution.

7.2.2 Three-dimensional differential operators

In Chebop2 [319] it is suggested to split two-dimensional differential operators via an SVD into a sum of one-dimensional operators, for which the matrices L can be computed as in Section 7.2.1. Such splittings can be generalized to higher dimensional settings via CP decompositions (2.4).

Let $\mathcal{A} \in \mathbb{R}^{(N_x+1)\times(N_y+1)\times(N_z+1)}$ denote the tensor with entries $\mathcal{A}_{a,b,c}$ given by the coefficients α_{abc} in Equation (7.2) for $0 \le a \le N_x,\ 0 \le b \le N_y,\ 0 \le c \le N_z$. Given a CP decompositon (2.4) of the form $\mathcal{A} = \sum_{r=1}^{R} \mathbf{a}^{(r)} \circ \mathbf{b}^{(r)} \circ \mathbf{c}^{(r)}$, we can rewrite the application of $\mathcal{L}$ to u as

$$\mathcal{L}u(x) = \sum_{r=1}^{R} \underbrace{\sum_{a=0}^{N_x} \mathbf{a}_a^{(r)} \frac{\partial^a}{\partial x^a}}_{=\mathcal{L}_r^{(x)}} \underbrace{\sum_{b=0}^{N_y} \mathbf{b}_b^{(r)} \frac{\partial^b}{\partial y^b}}_{=\mathcal{L}_r^{(y)}} \underbrace{\sum_{a=0}^{N_z} \mathbf{c}_c^{(r)} \frac{\partial^c}{\partial z^c}}_{=\mathcal{L}_r^{(z)}} u(x,y,z), \tag{7.7}$$

in terms of one-dimensional differential operators $\mathcal{L}_r^{(x)}, \mathcal{L}_r^{(y)}, \mathcal{L}_r^{(z)}$.

Let $L_r^{(x)} \in \mathbb{R}^{(n_1+1)\times(n_1+1)}$, $L_r^{(y)} \in \mathbb{R}^{(n_2+1)\times(n_2+1)}$, $L_r^{(z)} \in \mathbb{R}^{(n_3+1)\times(n_3+1)}$ denote the matrices L associated with the operators $\mathcal{L}_r^{(x)}, \mathcal{L}_r^{(y)}, \mathcal{L}_r^{(z)}$ as defined in Equation (7.6). Let u be a polynomial with coefficient tensor $\mathcal{U} \in \mathbb{R}^{(n_1+1)\times(n_2+1)\times(n_3+1)}$ satisfying Equation (7.4) with equality. Then

$$\mathcal{L}u(x,y,z) = \sum_{i=0}^{n_1}\sum_{j=0}^{n_2}\sum_{k=0}^{n_3} \mathcal{V}_{ijk} C_i^{(N_x)}(x) C_j^{(N_y)}(y) C_j^{(N_z)}(z),$$

with coefficient tensor

$$\mathcal{V} = \sum_{r=1}^{R} \mathcal{U} \times_1 L_r^{(x)} \times_2 L_r^{(y)} \times_3 L_r^{(z)}. \tag{7.8}$$

7.2.3 Generalization to non-constant coefficients

One-dimensional differential operators

We now consider the one-dimensional differential operator (7.5) with non-constant coefficients $\alpha_a : [-1,1] \to \mathbb{R}$. In this section, we define multiplication matrices to incorporate these coefficients in the discretization.

Let $u(x)$ be a polynomial of the form $u(x) = \sum_{k=0}^{n} u_k B_k(x)$, with basis functions $B_k(x)$ chosen as Chebyshev polynomials T_k or ultraspherical polynomials $C_k^{(\lambda)}$. To multiply $u(x)$ by a function $v : [-1,1] \to \mathbb{R}$, we approximate $v(x) \approx \tilde{v}(x) = \sum_{k=0}^{n} v_k B_k(x)$ by a polynomial in the same basis using Chebyshev interpolation [322] (and basis transformations).

Further, we also approximate the product $\tilde{v}(x)u(x) \approx z(x) = \sum_{k=0}^{n} z_k B_k(x)$ by a polynomial in the same basis. There exist so-called multiplication matrices [124, 246] depending on the coefficients $\mathbf{v} = (v_0, \dots, v_n)$, which map the coefficients $\mathbf{u} = (u_0, \dots, u_n)$ to the coefficients $\mathbf{z} = (z_0, \dots, z_n)$ such that $z(x)$ approximates $\tilde{v}(x)u(x)$. In the following, we summarize how these matrices are defined in [318] for both Chebyshev and ultraspherical bases. For the Chebyshev basis, we define $\mathbf{z} = M[\mathbf{v}]\mathbf{u}$ and the multiplication matrix $M[\mathbf{v}] \in \mathbb{R}^{(n+1)\times(n+1)}$ given by

$$
\begin{aligned}
M[\mathbf{v}] =& \frac{1}{2}\begin{pmatrix}
2v_0 & v_1 & v_2 & \dots & v_{n-1} & v_n \\
v_1 & 2v_0 & v_1 & \dots & v_{n-2} & v_{n-1} \\
v_2 & v_1 & 2v_0 & \dots & v_{n-3} & v_{n-2} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
v_{n-1} & v_{n-2} & v_{n-3} & \dots & 2v_0 & v_1 \\
v_n & v_{n-1} & v_{n-2} & \dots & v_1 & 2v_0
\end{pmatrix} \\
&+ \frac{1}{2}\begin{pmatrix}
0 & 0 & 0 & \dots & 0 & 0 & 0 \\
v_1 & v_2 & v_3 & \dots & v_{n-1} & v_n & 0 \\
v_2 & v_3 & v_4 & \dots & v_n & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
v_{n-1} & v_n & 0 & \dots & 0 & 0 & 0 \\
v_n & 0 & 0 & \dots & 0 & 0 & 0
\end{pmatrix}.
\end{aligned}
$$

For the $C^{(\lambda)}$ ultraspherical basis, we define $\mathbf{z} = M^{(\lambda)}[\mathbf{v}]\mathbf{u}$ and the multiplication matrix $M^{(\lambda)}[\mathbf{v}] \in \mathbb{R}^{(n+1)\times(n+1)}$ as $M^{(\lambda)}[\mathbf{v}] = \sum_{i=0}^{n} v_i M_i^{(\lambda)}$. The matrices $M_i^{(\lambda)} \in \mathbb{R}^{(n+1)\times(n+1)}$ are defined recursively for $i = 0, 1, \dots$ by

$$
(i+2)M_{i+2}^{(\lambda)} = 2(i+\lambda{+}1)N^{(\lambda)}M_{i+1}^{(\lambda)} - (i+2\lambda)M_i^{(\lambda)},
$$

with $M_0^{(\lambda)} = I$, $M_1^{(\lambda)} = 2\lambda N^{(\lambda)}$ and $N^{(\lambda)} \in \mathbb{R}^{(n+1)\times(n+1)}$ defined as

$$
N^{(\lambda)} = \begin{pmatrix}
0 & \frac{2\lambda}{2(\lambda+1)} & & & & & & \\
\frac{1}{2\lambda} & 0 & \frac{2\lambda+1}{2(\lambda+2)} & & & & & \\
0 & \frac{2}{2(\lambda+1)} & 0 & \frac{2\lambda+2}{2(\lambda+3)} & & & & \\
 & & \frac{2}{2(\lambda+2)} & 0 & \frac{2\lambda+3}{2(\lambda+4)} & & & \\
 & & & \ddots & \ddots & \ddots & & \\
 & & & & \frac{2}{2(\lambda+n-2)} & 0 & \frac{2\lambda+n-1}{2(\lambda+n)} & \\
 & & & & & \frac{2}{2(\lambda+n-1)} & 0 &
\end{pmatrix}.
$$

We now incorporate the non-constant coefficients $\alpha_a(x)$ into the coefficient mapping. Let $\mathbf{a}_a \in \mathbb{R}^{(n+1)}$ denote coefficients of polynomial approximations of $\alpha_a(x)$ in the Chebyshev basis for $a = 0$ and in $C^{(a)}$ ultraspherical basis for $a = 1, 2, \dots$. Analogous to

Equation (7.6), we approximate $\mathcal{L}u \approx \sum_{k=0}^{n} w_k T_k(x)$ by computing the coefficients $\mathbf{w} = (w_0, \dots, w_n)$ defined as

$$\begin{aligned}\mathbf{w} = L\mathbf{u} :=& (M^{(N)}[\mathbf{a}_N]D_N + S_{N-1}M^{(N-1)}[\mathbf{a}_{N-1}]D_{N-1} + \dots \\ &+ S_{N-1}\cdots S_1 M^{(1)}[\mathbf{a}_1]D_1 + S_{N-1}\cdots S_0 M[\mathbf{a}_0])\mathbf{u}.\end{aligned}$$

Remark 7.2. *The interpolation of v and the approximation of the product $\tilde{v}u$ introduce truncation errors. For sufficiently large n, these errors are close to machine precision.*

Three-dimensional differential operators

We now consider the three-dimensional differential operator $\mathcal{L}$ as defined in Equation (7.2) with non-constant coefficients. We proceed by splitting this operator into one-dimensional operators with non-constant coefficients.

Let the polynomial degree (n_1, n_2, n_3) be chosen sufficiently large to accurately approximate the coefficient functions $\alpha_{abc}(x, y, z)$ using tensorized Chebyshev interpolation in the form of Equation (7.4). Let $\mathcal{B}^{(abc)} \in \mathbb{R}^{(n_1+1)\times(n_2+1)\times(n_3+1)}$ denote the coefficient tensors corresponding to $\alpha_{abc}(x, y, z)$ for each multi-index abc. We define the tensor $\mathcal{A} \in \mathbb{R}^{(N_x+1)\times(n_1+1)\times(N_y+1)\times(n_2+1)\times(N_z+1)\times(n_3+1)}$ with entries

$$\mathcal{A}_{aibjck} = \mathcal{B}^{(abc)}_{ijk}$$

for $0 \le a \le N_x,\ 0 \le b \le N_y,\ 0 \le c \le N_z,\ 0 \le i \le n_1,\ 0 \le j \le n_2,\ 0 \le k \le n_3$. This approximates the differential operator in the sense that

$$\begin{aligned}&\mathcal{L}u(x, y, z) \approx \tilde{\mathcal{L}}u(x, y, z) := \\ &\sum_{a=0}^{N_x}\sum_{i=0}^{n_1}\sum_{b=0}^{N_y}\sum_{j=0}^{n_2}\sum_{c=0}^{N_z}\sum_{k=0}^{n_3} \mathcal{A}_{aibjck} T_i(x)T_j(y)T_k(z)\frac{\partial^{a+b+c}}{\partial x^a \partial y^b \partial z^c} u(x, y, z).\end{aligned}$$

We now reshape $\mathcal{A}$ into a tensor of order 3 in $\mathbb{R}^{(N_x+1)(n_1+1)\times(N_y+1)(n_2+1)\times(N_z+1)(n_3+1)}$ and compute a CP decomposition of the form

$$\mathcal{A} = \sum_{r=1}^{R} \mathbf{a}^{(r)} \circ \mathbf{b}^{(r)} \circ \mathbf{c}^{(r)},$$

where $\mathbf{a}^{(r)} \in \mathbb{R}^{(N_x+1)(n_1+1)}$, $\mathbf{b}^{(r)} \in \mathbb{R}^{(N_y+1)(n_2+1)}$, $\mathbf{c}^{(r)} \in \mathbb{R}^{(N_z+1)(n_3+1)}$. We reshape the vectors $\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r$ back into matrices in $\mathbf{A}^{(r)} \in \mathbb{R}^{(N_x+1)\times(n_1+1)}$, $\mathbf{B}^{(r)} \in \mathbb{R}^{(N_y+1)\times(n_2+1)}$ and $\mathbf{C}^{(r)} \in \mathbb{R}^{(N_z+1)\times(n_3+1)}$, respectively. Analogously to Equation (7.7), we now define one-dimensional differential operators

$$\mathcal{L}_r^{(x)} u(x) = \sum_{a=0}^{N_x} \left(\sum_{i=0}^{n_1} \mathbf{A}_{ai}^{(r)} T_i(x) \right) \frac{\partial^a}{\partial x^a} u(x),$$

$$\mathcal{L}_r^{(y)} u(y) = \sum_{b=0}^{N_y} \left(\sum_{j=0}^{n_2} \mathbf{B}_{bj}^{(r)} T_j(y) \right) \frac{\partial^b}{\partial y^b} u(y),$$

$$\mathcal{L}_r^{(z)} u(z) = \sum_{c=0}^{N_z} \left(\sum_{k=0}^{n_3} \mathbf{C}_{ck}^{(r)} T_k(z) \right) \frac{\partial^c}{\partial z^c} u(z),$$

which satisfy by construction

$$\tilde{\mathcal{L}} u(x,y,z) = \sum_{r=1}^{R} (\mathcal{L}_r^{(x)} \otimes \mathcal{L}_r^{(y)} \otimes \mathcal{L}_r^{(z)}) u(x,y,z), \tag{7.9}$$

where $\otimes$ denotes the tensor product for linear operators. Note that the terms of the form $\sum_{i=0}^{n_1} \mathbf{A}_{ai}^{(r)} T_i(x)$ are univaritate functions. So, each of the one-dimensional differential operators fits into the setting of Section 7.2.3 and we can obtain matrices $L_r^{(x)}, L_r^{(y)}, L_r^{(z)}$ as in Equation (7.8).

Remark 7.3. *In our numerical experiments we use using tensorlab [332] to obtain the CP decomposition from the full tensor $\mathcal{A}$. Since $\mathcal{A}$ is potentially very large, it might sometimes be necessary to rely on heuristic algorithms that do not explicitly require the full tensor.*

7.3 A spectral method for three-dimensional linear PDEs

In this section, we present how to compute approximate solutions for PDEs of the form (7.3). We again discretize the differential operator $\mathcal{L}$ as in Section 7.2. Additionally, we discretize the right hand side f using a truncated expansion with ultraspherical basis functions of the form

$$f(x,y,z) \approx \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} \sum_{k=0}^{n_3} \mathcal{F}_{ijk} C_i^{(N_x)}(x) C_j^{(N_y)}(y) C_k^{(N_z)}(z),$$

with coefficient tensor $\mathcal{F} \in \mathbb{R}^{(n_1+1)\times(n_2+1)\times(n_3+1)}$. The tensor $\mathcal{F}$ can be computed by applying suitable basis transformations to the coefficient tensor obtained from tensorized Chebyshev interpolation. The discretized PDE reads as tensor-valued linear system of the form

$$\sum_{r=1}^{R} \mathcal{U} \times_1 L_r^{(x)} \times_2 L_r^{(y)} \times_3 L_r^{(z)} = \mathcal{F}. \tag{7.10}$$

It remains to incorporate the boundary conditions.

7.3.1 Boundary condition discretization

Following the ideas of [319], we can discretize commonly used boundary conditions for three-dimensional PDEs on cubes as constraints of the form

$$\mathcal{U} \times_1 B_1 = \mathcal{G}_1, \quad \mathcal{U} \times_2 B_2 = \mathcal{G}_2, \quad \mathcal{U} \times_3 B_3 = \mathcal{G}_3, \tag{7.11}$$

where the matrices $B_1 \in \mathbb{R}^{N_x \times (n_1+1)}, B_2 \in \mathbb{R}^{N_y \times (n_2+1)}, B_3 \in \mathbb{R}^{N_z \times (n_3+1)}$ have linearly independent rows and $\mathcal{G}_1 \in \mathbb{R}^{N_x \times (n_2+1) \times (n_3+1)}, \mathcal{G}_2 \in \mathbb{R}^{(n_1+1) \times N_y \times (n_3+1)}, \mathcal{G}_3 \in \mathbb{R}^{(n_1+1) \times (n_2+1) \times N_z}$. We present two examples of how constraints of the form (7.11) can be derived. We want to emphasize that the discretized boundary conditions need to satisfy compatibility constraints as in [318, Section 6.4].

Dirichlet conditions

We consider the Dirichlet boundary condition $u(1, y, z) = h(y, z)$ for a given function $h : [-1, 1] \to \mathbb{R}$. We approximate the function h using bivariate Chebyshev interpolation in $(n_2{+}1) \times (n_3{+}1)$ points. Let $H \in \mathbb{R}^{(n_2+1) \times (n_3+1)}$ denote the corresponding coefficient matrix of h. We can enforce that the solution u given in Chebyshev basis (7.4) coincides with the interpolant of h by demanding that

$$u(1, y, z) = \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} \sum_{k=0}^{n_3} \mathcal{U}_{ijk} T_i(1) T_j(y) T_k(z) = \sum_{j=0}^{n_2} \sum_{k=0}^{n_3} H_{jk} T_j(y) T_k(z). \tag{7.12}$$

This can be equivalently written as $U \times_1 B_1 = \mathcal{G}_1$ with $B_1 = (T_0(1), \ldots, T_{n_1}(1))$ and $(G_1)_{0,0:n_2,0:n_3} = H$.

We now consider an additional Dirichlet boundary condition $u(-1, y, z) = \tilde{h}$. Again, we compute the coefficient matrix $\tilde{H}$ corresponding to $\tilde{h}$. Analogously to Equation (7.12), we enforce that $u(-1, y, z)$ coincides with the interpolant of $\tilde{h}$. We can express both conditions simultaneously in the form of (7.11) by defining

$$B_1 = \begin{pmatrix} T_0(-1) & T_1(-1) & \ldots & T_n(-1) \\ T_0(1) & T_1(1) & \ldots & T_n(1) \end{pmatrix},$$

and $(G_1)_{0,0:n_2,0:n_3} = \tilde{H}, \ (G_1)_{1,0:n_2,0:n_3} = H$.

Mixed Dirchlet and Neumann conditions

We consider one example of mixed boundary conditions with Neumann boundary conditions on the right side of the cube $[-1,1]^3$ and Dirichlet boundary conditions on all other sides. The Neumann boundary condition is given by $\frac{\partial}{\partial x}u(1,y,z) = h(y,z)$ for a given function h and $u \in \mathbb{P}_{n_1,n_2,n_3}$ represented in Chebyshev basis (7.4). As in the Dirichlet case, we use bivariate Chebyshev interpolation h to obtain the coefficient matrix $H \in \mathbb{R}^{(n_2+1)\times(n_3+1)}$. We now demand that

$$\frac{\partial}{\partial x}u(1,y,z) = \sum_{i=0}^{n_1}\sum_{j=0}^{n_2}\sum_{k=0}^{n_3}\mathcal{U}_{ijk}T_i'(1)T_j(y)T_k(z) = \sum_{j=0}^{n_2}\sum_{k=0}^{n_3}H_{jk}T_j(y)T_k(z), \tag{7.13}$$

where $T_i'(1) = i^2$ for $i = 0,\dots,n$ [322]. Equation (7.13) can be expressed in the form of (7.11) with

$$B_1 = (T_0'(1),\dots,T_{n_1}'(1)) \quad \text{and} \quad (G_1)_{0,0:n_2,0:n_3} = H.$$

The Dirichlet boundary conditions on the left side can be included in $B_1, \mathcal{G}_1$ as in Section 7.3.1.

7.3.2 Incorporating the boundary conditions

We need to incorporate the discretized boundary conditions (7.11) into the discretized PDE (7.10) to obtain the unique solution $\mathcal{U}$. Following the ideas of [319, Section 6], we compute $\mathcal{U}$ by substituting (7.11) into (7.10).

Since B_1, B_2, B_3 have linearly independent rows, we can assume without loss of generality that

$$(B_1)_{0:(N_x-1),0:(N_x-1)} = I, \quad (B_2)_{0:(N_y-1),0:(N_y-1)} = I, \quad (B_3)_{(0:(N_z-1),0:(N_z-1)} = I. \tag{7.14}$$

For $r = 1,\dots,R$, we rewrite the boundary conditions as

$$\begin{aligned}
-\mathcal{U} \times_1 (L_r^{(x)})_{0:n_1,0:(N_x-1)}B_1 &= -\mathcal{G}_1 \times_1 (L_r^{(x)})_{0:n_1,0:(N_x-1)},\\
-\mathcal{U} \times_2 (L_r^{(y)})_{0:n_2,0:(N_y-1)}B_2 &= -\mathcal{G}_2 \times_2 (L_r^{(y)})_{0:n_2,0:(N_y-1)},\\
-\mathcal{U} \times_3 (L_r^{(z)})_{0:n_3,0:(N_z-1)}B_3 &= -\mathcal{G}_3 \times_3 (L_r^{(z)})_{0:n_3,0:(N_z-1)}.
\end{aligned}$$

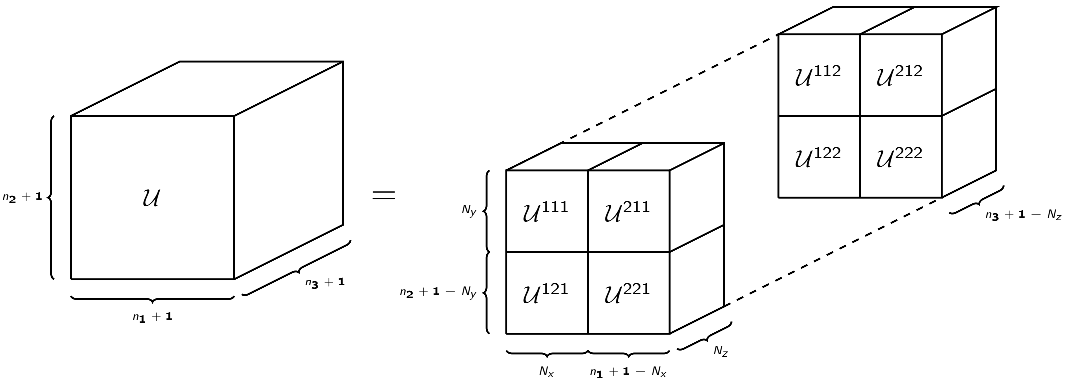


Figure 7.1 – Visualization of the block decomposition of $\mathcal{U} \in \mathbb{R}^{(n_1+1)\times(n_2+1)\times(n_3+1)}$ with block $\mathcal{U}^{111} \in \mathbb{R}^{N_x \times N_y \times N_z}$.

Substituting these modified boundary conditions into the discretized PDE (7.10) leads to

$$\begin{aligned}
&\sum_{r=1}^{R} \mathcal{U} \times_1 \underbrace{(L_r^{(x)} - (L_r^{(x)})_{0:n_1,0:(N_x-1)} B_1)}_{=\tilde{L}_r^{(x)}} \times_2 \underbrace{(L_r^{(y)} - (L_r^{(y)})_{0:n_2,0:(N_y-1)} B_2)}_{=\tilde{L}_r^{(y)}} \\
&\qquad \times_3 \underbrace{(L_r^{(z)} - (L_r^{(z)})_{0:n_3,0:(N_z-1)} B_3)}_{=\tilde{L}_r^{(z)}} = \\
&\quad \mathcal{F} - \sum_{r=1}^{R} \mathcal{G}_1 \times_1 (L_r^{(x)})_{(0:n_1,0:(N_x-1)} \times_2 L_r^{(y)} \times_3 L_r^{(z)} \\
&\quad - \sum_{r=1}^{R} \mathcal{G}_2 \times_1 (L_r^{(x)} - (L_r^{(x)})_{0:n_1,0:(N_x-1)} B_1) \times_2 (L_r^{(y)})_{0:n_2,0:\,(N_y-1)} \times_3 L_r^{(z)} \\
&\quad - \sum_{r=1}^{R} \mathcal{G}_3 \times_1 (L_r^{(x)} - (L_r^{(x)})_{0:n_1,0:(N_x-1)} B_1) \times_2 (L_r^{(y)} - (L_r^{(y)})_{0:n_2,0:(N_y-1)} B_2) \\
&\qquad \times_3 (L_r^{(z)})_{0:n_3,0:(N_z-1)},
\end{aligned}$$

where we denote the right hand side by $\tilde{\mathcal{F}}$. Observe that the first N_x, N_y, N_z columns of $\tilde{L}_r^{(x)}, \tilde{L}_r^{(y)}, \tilde{L}_r^{(z)}$ are zero due to assumption (7.14). Let $\mathcal{U}$ be decomposed into tensor blocks as described in Figure 7.1. The system after substitution uniquely determines the block $\mathcal{U}^{222} = \mathcal{U}_{N_x:n_1,N_y:n_2,N_z:n_3}$, which can be written as

$$\begin{aligned}
&\sum_{r=1}^{R} \mathcal{U}^{222} \times_1 \underbrace{(\tilde{L}_r^{(x)})_{0:(n_1-N_x),N_x:n_1}}_{=\hat{L}_r^{(x)}} \times_2 \underbrace{(\tilde{L}_r^{(y)})_{0:(n_2-N_y),N_y:n_2}}_{=\hat{L}_r^{(y)}} \times_3 \underbrace{(\tilde{L}_r^{(z)})_{0:(n_3-N_z),N_z:n_3}}_{=\hat{L}_r^{(z)}} \\
&\quad = \underbrace{\tilde{\mathcal{F}}_{0:(n_1-N_x),0:(n_2-N_y),0:(n_3-N_z)}}_{=\hat{\mathcal{F}}}.
\end{aligned} \tag{7.15}$$

The remaining blocks can be reconstructed from the discretized boundary conditions (7.11) as

$$\begin{aligned}
\mathcal{U}^{122} &= \mathcal{G}_1 - \mathcal{U}^{222} \times_1 B_1, & \mathcal{U}^{212} &= \mathcal{G}_2 - \mathcal{U}^{222} \times_2 B_2,\\
\mathcal{U}^{221} &= \mathcal{G}_3 - \mathcal{U}^{222} \times_3 B_3, & \mathcal{U}^{112} &= \mathcal{G}_1 - \mathcal{U}^{212} \times_1 B_1,\\
\mathcal{U}^{121} &= \mathcal{G}_1 - \mathcal{U}^{221} \times_1 B_1, & \mathcal{U}^{211} &= \mathcal{G}_3 - \mathcal{U}^{212} \times_3 B_3,\\
\mathcal{U}^{111} &= \mathcal{G}_1 - \mathcal{U}^{211} \times_1 B_1.
\end{aligned}$$

7.3.3 Solving tensor-valued linear systems

The computation of $\mathcal{U}$ requires the solution of the unconstrained tensor-valued linear system (7.15) of the form

$$\sum_{r=1}^{R} \mathcal{U}^{222} \times_1 \hat{L}_r^{(x)} \times_2 \hat{L}_r^{(y)} \times_3 \hat{L}_r^{(z)} = \hat{\mathcal{F}}. \tag{7.16}$$

This system can be solved by reshaping the tensor-valued linear system into a vector-valued linear system of the form

$$\left(\sum_{r=1}^{R} \hat{L}_r^{(z)} \otimes \hat{L}_r^{(y)} \otimes \hat{L}_r^{(x)} \right) \mathsf{vec}(\mathcal{U}^{222}) = \mathsf{vec}(\hat{\mathcal{F}}). \tag{7.17}$$

For certain PDEs we can transform the system (7.16) into a Laplace-like equation

$$\mathcal{U}^{222} \times_1 U + \mathcal{U}^{222} \times_2 V + \mathcal{U}^{222} \times_3 W = \breve{\mathcal{F}}, \tag{7.18}$$

with matrices $U \in \mathbb{R}^{(n_1+1-N_x)\times(n_1+1-N_x)}$, $V \in \mathbb{R}^{(n_2+1-N_y)\times(n_2+1-N_y)}$, $W \in \mathbb{R}^{(n_3+1-N_z)\times(n_3+1-N_z)}$ and tensor $\breve{\mathcal{F}} \in \mathbb{R}^{(n_1+1-N_x)\times(n_2+1-N_y)\times(n_3+1-N_z)}$. For instance, this can be achieved for $N_x = N_y = N_z$, $n_1 = n_2 = n_3$ and $B_1 = B_2 = B_3$, when there exists a CP decomposition of the tensor $\mathcal{A}$ defined in Section 7.2 with symmetry constraints [61] of the form

$$\mathcal{A} = \mathbf{a} \circ \mathbf{v} \circ \mathbf{w} + \mathbf{u} \circ \mathbf{b} \circ \mathbf{w} + \mathbf{u} \circ \mathbf{v} \circ \mathbf{c}. \tag{7.19}$$

Then Equation (7.16) is equivalent to the Laplace-like equation (7.18) with

$$\begin{aligned}
&U = (\hat{L}_1^{(y)})^{-1} \hat{L}_1^{(x)}, \quad V = (\hat{L}_2^{(x)})^{-1} \hat{L}_2^{(y)}, \quad W = (\hat{L}_3^{(x)})^{-1} \hat{L}_3^{(z)},\\
&\breve{\mathcal{F}} = \hat{\mathcal{F}} \times_1 (\hat{L}_1^{(y)})^{-1} \times_2 (\hat{L}_2^{(x)})^{-1} \times_3 (\hat{L}_3^{(x)})^{-1}.
\end{aligned} \tag{7.20}$$

To solve Laplace-like equations (7.18), we apply the recursive blocked algorithm developed in [69]. It transforms the matrices U, V, W into quasi-triangular form by computing Schur decompositions. Block decompositions for the quasi-triangular matrices reveal an equivalent system of Laplace-like equations with smaller matrices. We apply this observation recursively, until we can solve the small Laplace-like equations efficiently by reshaping. This yields the blocked recursive algorithm. For $n = n_1 = n_2 = n_3$ this approach has a theoretical runtime of $\mathcal{O}(n^4)$ operations. In Section 7.4.1, we demonstrate that this recursive blocked algorithm is much faster than directly reshaping the tensor-valued linear system.

Remark 7.4. *For general PDEs with the same type of boundary conditions in each mode, we can obtain a CP decomposition of the form* (7.19) *if the differential operator does not contain mixed derivatives and if all coefficients* α_{abc} *only depend on the variable corresponding to the mode in which their corresponding derivative acts. This includes, for instance, differential operators of the form*

$$\alpha_1(x) + \alpha_2(x)\frac{\partial}{\partial x} + \alpha_3(x)\frac{\partial^2}{\partial x^2} + \alpha_4(y)\frac{\partial}{\partial y} + \alpha_5(y)\frac{\partial^2}{\partial y^2} + \alpha_6(z)\frac{\partial}{\partial z} + \alpha_7(z)\frac{\partial^2}{\partial z^2},$$

with univariate coefficient functions $\alpha_i : [-1, 1] \to \mathbb{R}$.

Remark 7.5. *So far, we introduced the global spectral method for fixed polynomial degrees* (n_1, n_2, n_3). *In order to heuristically determine if the solution is accurate, we can analyze the residual of* (7.10) *and the decay of the coefficients in* $\mathcal{U}$ *[16]. This can be used to adaptively increase* (n_1, n_2, n_3) *until the solution is accurate.*

Remark 7.6. *In general, it is not recommended to explicitly invert the matrices in* (7.20). *The inversion can be avoided completely by extending the recursive solver in [69] similar to how recursive solvers for Sylvester equations can be extended to solve generalized Sylvester equations.*

7.4 Numerical results

All numerical experiments* in this section were performed in MATLAB R2018b on a Lenovo Thinkpad T480s with Intel Core i7-8650U CPU and 15.4 GiB RAM.

7.4.1 Runtime comparison

We consider Poissons's equation $\Delta u = f$ with homogeneous zero Dirichlet boundary conditions. In Table 7.1, we compare our global spectral method to the nested alternating direction implicit method (NADIM) proposed in [125, Section 5]. NADIM relies on solving two-dimensional Sylvester equations recursively on three levels. On each level an iterative algorithm is used, which leads to a large total number of iterations.

---

*The MATLAB code to reproduce these results is available from https://github.com/cstroessner/SpectralMethod3D.

	$n = 10$		$n = 30$		$n = 50$		$n = 150$	
	Time	Error	Time	Error	Time	Error	Time	Error
NADIM	21.8	$1.47 \cdot 10^{-5}$	742	$2.80 \cdot 10^{-8}$	-	-	-	-
reshape	0.019	$1.55 \cdot 10^{-5}$	2.40	$1.22 \cdot 10^{-15}$	102	$1.22 \cdot 10^{-15}$	-	-
recursive	0.054	$1.55 \cdot 10^{-5}$	0.10	$3.12 \cdot 10^{-13}$	0.41	$8.20 \cdot 10^{-13}$	10.9	$1.15 \cdot 10^{-10}$

Table 7.1 – Comparison of algorithms to solve Poisson's equation with known solution $u^*(x, y, z) = \sin(\pi x)\sin(\pi y)\sin(\pi z)$, from which the right hand side f is explicitly computed. To compute solutions $u \in \mathbb{P}_{n,n,n}$ with our global spectral method, we compare reshaping (7.15) and solving (7.17) with backslash in MATLAB (reshape) to transforming (7.15) into a Laplace-like equation (7.18) and solving with the blocked recursive solver [69] (recursive). Additionally, we compare to NADIM [125]. For various n, we measure the runtime in seconds and we estimate $||u - u^*||_\infty$ by sampling 1 000 random points.

We observe that even though NADIM has an asymptotic runtime of $\mathcal{O}(n^3(\log(n))^3)$ [125], it is the slowest algorithm in our setting and can not handle $n > 30$ in a reasonable amount of time. The asymptotic runtime of the recursive algorithm is slower with $\mathcal{O}(n^4)$, but in our experiments it is the fastest method and it can handle values of up to $n = 150$ in less than 11 seconds. For $n = 50$ solving the reshaped system (7.17) with backslash leads to an error of order 10^{-15}, whereas the recursive algorithm only achieves an error of order 10^{-13}. While the recursive approach is able to solve much larger systems, it is slightly more sensitive to numerical rounding errors.

7.4.2 Stationary problems

Helmholtz problems

We consider the Helmholtz equation on $[-1, 1]^3$ with non-homogeneous Dirichlet boundary conditions as in [160, 319], which arises for instance in the context of three-dimensional wave equations in acoustics [336] and seismic-imaging [261]. It is defined as

$$\Delta u(x, y, z) + \kappa^2 u(x, y, z) = f(x, y, z),$$

with coefficient $\kappa \in \mathbb{R}$. For this differential operator, we define the tensor $\mathcal{A}$ as in Section 7.2.2. We observe that a CP decomposition in the form (7.19) is given by

$$\mathcal{A} = \begin{pmatrix} \kappa^2 \\ 0 \\ 1 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \circ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \circ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{7.21}$$

Hence, we can derive a Laplace-like structure (7.18) for Equation (7.15) and employ the recursive solver.

In the following, we employ the global spectral method for the Helmholtz equation in [56, Section 5.3] given by

$$\Delta u(x,y,z) + \kappa(x)^2 u(x,y,z) = f(x,y,z), \tag{7.22}$$

with function $\kappa(x) = \gamma_1 - \gamma_2 \cos(\pi\gamma_3 x/2)$ and scalar coefficients $\gamma_1, \gamma_2, \gamma_3 \in \mathbb{R}$. The right hand side f and the Dirichlet boundary conditions are computed explicitly from the solution

$$u^*(x,y,z) = \exp(-\kappa(x)/\gamma_3)\cos(\pi\gamma_1 y/2)\cos(\pi\gamma_2 z/2). \tag{7.23}$$

In order to incorporate the coefficient function $\kappa(x)$, we compute one-dimensional differential operators as in Equation (7.7) from the CP-decomposition (7.21). We then set $\mathcal{L}_1^{(x)} u(x) = \kappa(x)^2 u(x) + \frac{\partial^2}{\partial x^2} u(x)$ and discretize this operator as described in Section 7.2.3. The resulting equation (7.15) can still be transformed into a Laplace-like equation (7.18) by setting $U, V, W, \breve{\mathcal{F}}$ as defined in (7.20).

In Figure 7.2, we depict how well the solution $u \in \mathbb{P}_{n,n,n}$ of our global spectral method approximates u^*. We compare to the trivariate Chebyshev interpolant $\tilde{u}$ of u^* that is close to the best approximation of u in $\mathbb{P}_{n,n,n}$ and converges quickly [100, 322]. For $n < 80$, the solution u and the interpolant $\tilde{u}$ almost coincide. For $n > 80$, the error $||u - u^*||_\infty$ stagnates, whilst the interpolation error $||\tilde{u} - u^*||_\infty$ continues to decrease further before reaching a plateau close to machine precision. This discrepancy is caused by the recursive blocked solver.

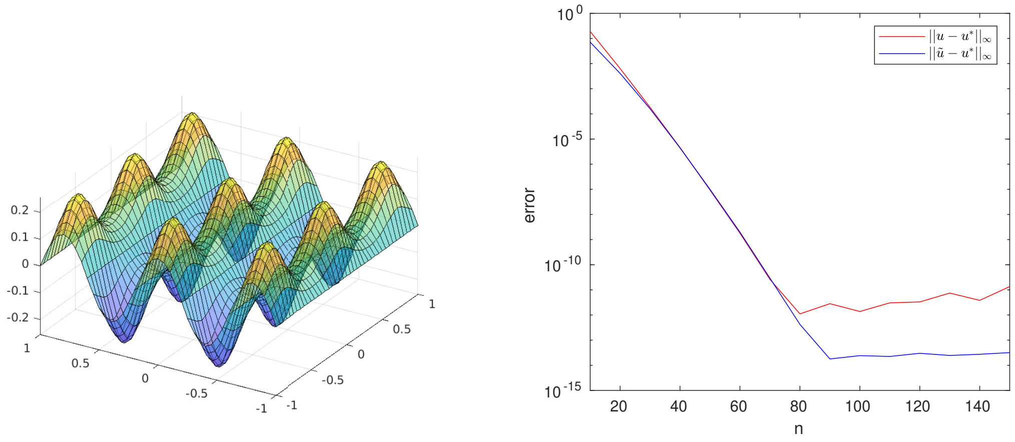


Figure 7.2 – Left: We plot $u^*(x,y,z)$ as defined in (7.23) for $(\gamma_1,\gamma_2,\gamma_3) = (5,3,5)$ and fixed $z = 1/4$. Right: We solve the Helmholtz equation (7.22) for $(\gamma_1,\gamma_2,\gamma_3) = (5,3,5)$ with known solution u^* (7.23) via our global spectral method to obtain the solution $u \in \mathbb{P}_{n,n,n}$ for various n. Further, we compute the trivariate Chebyshev interpolant $\tilde{u} \in \mathbb{P}_{n,n,n}$ of u^*. For each n, we estimate $||u - u^*||_\infty$ and $||\tilde{u} - u^*||_\infty$ using 1 000 sample points (evaluation error).

Remark 7.7. *The derivation of the Laplace-like system from the CP decomposition can be extended from the Helmholtz equation to convection diffusion problems of the form*

$$-\nu \triangle u + \xi^T \bigtriangledown u = f,$$

with $\nu \in R$ *and* $\xi \in \mathbb{R}^3$ *as studied in [24, 70, 340]. For these problems we define the CP-decomposition*

$$\mathcal{A} = \begin{pmatrix} 0 \\ \xi_1 \\ -\nu \end{pmatrix} \circ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \circ \begin{pmatrix} 0 \\ \xi_2 \\ -\nu \end{pmatrix} \circ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \circ \begin{pmatrix} 0 \\ \xi_3 \\ -\nu \end{pmatrix}.$$

Diffusion problems with separable coefficient

Diffusion problems of the form

$$-\bigtriangledown \cdot (a(x,y,z) \bigtriangledown u(x,y,z)) = f(x,y,z), \tag{7.24}$$

with a separable coefficient $a(x,y,z) = a_1(x)a_2(y)a_3(z)$ defined by univariate functions $a_1, a_2, a_3 : [-1,1] \to \mathbb{R}$, are not directly given as linear partial differential operator of the form (7.2). We can, however, decompose the three-dimensional differential operator into a sum of three differential operators similar to (7.7) as

$$\bigtriangledown \cdot (a(x,y,z) \bigtriangledown u(x,y,z)) = \\ \Big(\underbrace{\frac{\partial}{\partial x}\Big(a(x,y,z)\frac{\partial}{\partial x}\Big)}_{=\tilde{\mathcal{L}}_1^{(x)}} + \underbrace{\frac{\partial}{\partial y}\Big(a(x,y,z)\frac{\partial}{\partial y}\Big)}_{=\tilde{\mathcal{L}}_2^{(y)}} + \underbrace{\frac{\partial}{\partial z}\Big(a(x,y,z)\frac{\partial}{\partial z}\Big)}_{=\tilde{\mathcal{L}}_3^{(z)}} \Big) u(x,y,z).$$

Applying the differential operator $\tilde{\mathcal{L}}_1^{(x)}$ to a polynomial $u \in \mathbb{P}_{n,n,n}$ can be written analogously to (7.8) as

$$\mathcal{V} = \mathcal{U} \times_1 \underbrace{S_1 D_1 S_0^{-1} M^{(1)}[\mathbf{a}_1] D_1}_{=L_1^{(x)}} \times_2 \underbrace{M^{(2)}[\mathbf{a}_2] S_1 S_0}_{=L_1^{(y)}} \times_3 \underbrace{M^{(2)}[\mathbf{a}_3] S_1 S_0}_{=L_1^{(y)}}$$

where $\mathbf{a}_1 \in \mathbb{R}^{n_1+1}, \mathbf{a}_2 \in \mathbb{R}^{n_2+1}, \mathbf{a}_3 \in \mathbb{R}^{n_3+1}$ denote the coefficients for univariate Chebyshev interpolation of a_1, a_2, a_3 as in Section 7.2.3. The multiplication matrices $M[\mathbf{a}]$, differentiation matrices D_1 and transformation matrices S_0, S_1 are defined as in Section 7.2. We obtain analogous discretizations for $\tilde{\mathcal{L}}_2^{(y)}$ and $\tilde{\mathcal{L}}_3^{(z)}$. Observe that the discretization has the same symmetric structure as the CP decomposition (7.19). Thus, we can transform (7.15) to a Laplace-like equation (7.18) by defining $U, V, W, \breve{\mathcal{F}}$ as in (7.20) and use the recursive solver.

Diffusion problems with higher rank coefficient

Most diffusion problems (7.24) arising in the study of groundwater flow and uncertainty quantification [75, 133, 220, 287, 316, 328] do not have a rank-1 coefficient $a(x, y, z)$. The coefficient is often given by a truncated expansion as a sum of separable functions of the form

$$a(x, y, z) = \sum_{r=1}^{R_a} a_1^{(r)}(x) a_2^{(r)}(y) a_3^{(r)}(z),$$

with $R_a > 1$ and univariate functions $a_1^{(r)}, a_2^{(r)}, a_3^{(r)} : [-1, 1] \to \mathbb{R}$. The PDE (7.24) can now be written as

$$-\sum_{r=1}^{R_a} \bigtriangledown \cdot (a_1^{(r)}(x) a_2^{(r)}(y) a_3^{(r)}(z) \bigtriangledown u(x, y, z)) = f(x, y, z). \tag{7.25}$$

Following the ideas in Section 7.4.2, we can discretize (7.24) for each separable function $a_1^{(r)}(x) a_2^{(r)}(y) a_3^{(r)}(z)$. Adding these discretizations yields a discretization of the form (7.10) with $R = 3R_a$. Since $R > 3$, we can not find a Laplace-like formulation of (7.15) and we can not use the recursive solver.

We can, however, use preconditioned GMRES [137] to compute solutions of (7.16) seen as tensor-valued linear system. Throughout this work, we restart GMRES every 15 iterations. As preconditoner we employ the recursive solver to solve Equation (7.15) for a discretization of the same diffusion problem (7.25) with the coefficient $a(x, y, z)$ replaced by a separable coefficient $b(x, y, z)$. This can be seen as effectively solving the system

$$\left(\sum_{r=1}^{3} \hat{L}_r^{(z)}[b] \otimes \hat{L}_r^{(y)}[b] \otimes \hat{L}_r^{(x)}[b]\right)^{-1} \left(\sum_{r=1}^{R} \hat{L}_r^{(z)}[a] \otimes \hat{L}_r^{(y)}[a] \otimes \hat{L}_r^{(x)}[a]\right) \mathrm{vec}(\mathcal{U}^{222}) = \left(\sum_{r=1}^{3} \hat{L}_r^{(z)}[b] \otimes \hat{L}_r^{(y)}[b] \otimes \hat{L}_r^{(x)}[b]\right)^{-1} \mathrm{vec}(\hat{\mathcal{F}}),$$

where $\hat{L}_r^{(x)}[a], \hat{L}_r^{(y)}[a], \hat{L}_r^{(z)}[a]$ and $\hat{L}_r^{(x)}[b], \hat{L}_r^{(y)}[b], \hat{L}_r^{(z)}[b]$ denote the matrices in (7.16) based on discretizations of the PDE with coefficient a and b respectively. The application of the inverse of $\sum_{r=1}^{3} \hat{L}_r^{(z)}[b] \otimes \hat{L}_r^{(y)}[b] \otimes \hat{L}_r^{(x)}[b]$ can be computed by solving a Laplace-like equation (7.18).

From now on, we consider the rank-2 coefficient $a(x, y, z) = (1 + x^2)(1 + y^2)(1 + z^2) + \exp(x + y + z)$. We use our global spectral method with $n = n_1 = n_2 = n_3 = 30$ to solve the diffusion problem (7.25) with known solution $u^*(x, y, z) = \sin(\pi x) \sin(\pi y) \sin(\pi z)$, from which we explicitly compute the right hand side f and Dirichlet boundary conditions. In Figure 7.3, we display the convergence rates for GMRES with preconditioners based on the constant coefficient $b_1(x, y, z) = ||a||_{\mathcal{L}^2}$ and the separable coefficient $b_2(x, y, z) =$

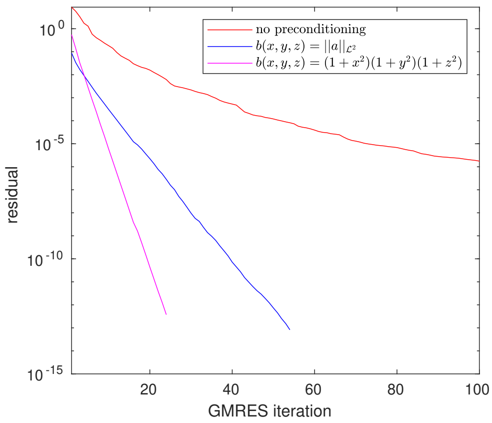


Figure 7.3 – Convergence rate of preconditioned GMRES for solving Equation (7.15) for the Diffusion problem (7.25) with coefficient $a(x,y,z) = (1+x^2)(1+y^2)(1+z^2)+\exp(x+y+z)$. We compare no preconditioning and preconditioning with a separable coefficient $b(x,y,z)$ as described in Section 7.4.2.

$(1+x^2)(1+y^2)(1+z^2)$. Both preconditioners yield solutions u such that the error $||u-u^*||_\infty$ is close to machine precision. Solving with b_1 takes 4.87 seconds. For the separable coefficient b_2, fewer iterations are necessary and the computation only takes 2.14 seconds. For comparison, reshaping as in Section 7.4.1 would take 107 seconds.

Helmholtz equation with non-constant coefficients

Next, we study a variable coefficient Helmholtz equation as in [319, Example 2]. We consider the PDE

$$\Delta u(x,y,z) + \kappa(x,y,z)u(x,y,z) = f(x,y,z), \tag{7.26}$$

with function $\kappa(x,y,z) = \sqrt{x+y+z+42}$. For $n = n_1 = n_2 = n_3 = 30$, we compute a discretization (7.10) by computing an approximate CP decomposition of $\mathcal{A}$ with $R = 10$ as defined as in Section 7.2.3 using tensorlab [332]. In 2.08 seconds, we obtain a CP decomposition with error $2.04 \cdot 10^{-9}$ in the uniform norm. This leads to a discretized PDE, for which we solve Equation (7.15) with preconditioned GMRES with restarting as in Section (7.4.2). In the preconditioner we solve Equation (7.15) for the Helmholtz equation (7.22) with constant coefficient $||\kappa||_{\mathcal{L}^2}$.

When the right hand side and the Dirichlet boundary conditions are chosen to match the known solution $u^*(x,y,z) = \sin(\pi x)\sin(\pi y)\sin(\pi z)$, solving with preconditioned GMRES takes 1.04 seconds. The computed solution u satisfies $||u-u^*||_\infty \approx 2.38 \cdot 10^{-10}$, where we estimate the uniform norm using 1 000 sample points.

Remark 7.8. *To obtain very high accuracy in the PDE solution, we would need a very accurate CP decomposition. The efficient computation of accurate CP decompositions is*

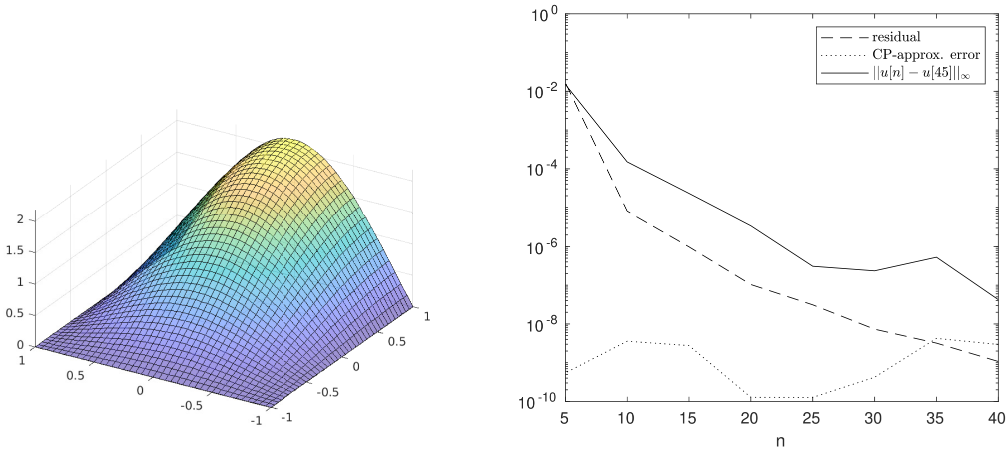


Figure 7.4 – Left: Let $u[n] \in \mathbb{P}_{n,n,n}$ denote the solution computed for the Helmholtz equation described in Section 7.4.2. We plot $u[45](x, y, 1/4)$. Right: For various n we plot $||u[n] - u[45]||_\infty$ estimated at 1 000 sample points. Additionally, we plot the combined residual defined as maximum of the residual of (7.10) and the residuals of (7.11) in the uniform norm. Further, we plot the error of the CP approximation of $\mathcal{A}$ in the uniform norm.

still subject to research [293]. Here, we could avoid computing a CP decompositoin of $\mathcal{A} \in \mathbb{R}^{93\times93\times93}$ by instead approximating the operators Δ and $\kappa(x, y, z)\mathcal{I}$ separately, where $\mathcal{I}$ denotes the identity operator. Discretizing $\kappa(x, y, z)\mathcal{I}$ only involves a CP decomposition of a tensor in $\mathbb{R}^{31\times31\times31}$, which can be computed for $R = 7$ in only 0.09 *seconds with error* $1.93 \cdot 10^{-11}$ *in the uniform norm. The two resulting discretizations in the form of* (7.9) *can be added to obtain a discretization of $\Delta + \kappa(x, y, z)\mathcal{I}$.*

Helmholtz equation with unknown solution

As a final stationary problem, we consider the Helmholtz equation (7.26) with non-constant coefficient $\kappa(x, y, z) = \sqrt{x + y + z + 42}$ and $f(x, y, z) = 1$. We use mixed boundary conditions with zero Neumann boundary conditions on the right and zero Dirichlet boundary conditions on all other sides as in Section 7.3.1. As in Section 7.4.2, we compute a CP decomposition of $\mathcal{A}$ and solve using preconditioned GMRES. In Figure 7.4, we display the computed solution and the error decay for different polynomial degrees. We observe that the residual decays when the polynomial degree is increased, which indicates that the solutions become more accurate.

7.4.3 Time-dependent problems

In this section, we introduce an implicit Euler scheme to solve parabolic PDEs of the form

$$\frac{\partial}{\partial t}u(x,y,z,t)+\mathcal{L}u(x,y,z,t)=0,$$

where $\mathcal{L}$ is a linear partial differential operator acting only on the spatial variables x,y,z. The system is complemented with boundary conditions. We are interested in the time evolution starting from a given initial function $u(x,y,z,0)=u_0(x,y,z)$. We discretize the equation in time using the uniform step length h. For each $\tau=1,2,\ldots$, we compute $u_\tau(x,y,z)\approx u(x,y,z,\tau h)$ as solution of the stationary linear PDE

$$(\mathcal{I}-h\mathcal{L})u_{\tau+1}=u_\tau, \tag{7.27}$$

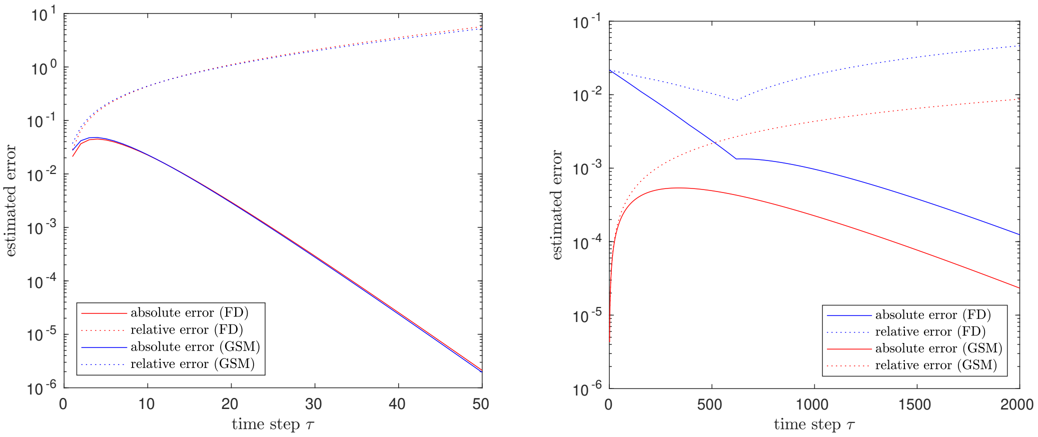
We approximate each function u_τ by a polynomial of the form (7.4) represented by the coefficient tensor $\mathcal{U}_\tau$. The initial $\mathcal{U}_0$ is computed via tensorized Chebyshev interpolation. For $\tau=1,2,\ldots$, we obtain $\mathcal{U}_\tau$ by applying our spectral method to solve Equation (7.27). We discretize the operator $(\mathcal{I}-h\mathcal{L})$ directly like the Helmholtz equation in Section 7.4.2. This allows us to employ the recursive solver. Note that the right hand side is represented in terms of an ultraspherical basis. Hence, the computation of $\mathcal{U}_\tau$ requires multiplying $\mathcal{U}_{\tau-1}$ with appropriate basis transformation matrices.

We demonstrate this implicit Euler scheme for the parabolic PDE studied in [257, Section 6.1]. The function $u^*(x,y,z,t)=\exp(-3\pi^2 t)\sin(\pi x)\sin(\pi y)\sin(\pi z)$ satisfies the parabolic PDE

$$\frac{\partial}{\partial t}u+\Delta u=0,$$

on the domain $[-1,1]^3$ with homogeneous Dirichlet boundary conditions. We use an implicit Euler scheme for $u_0(x,y,z)=\sin(\pi x)\sin(\pi y)\sin(\pi z)$ and compare our global spectral method to a finite difference method [303]. In each timestep of the finite difference method, we solve a linear system with a sparse Kronecker-structured finite difference matrix using the backslash operator in MATLAB.

The time evolution of the errors is displayed in Figure 7.5. We observe that for $n=20$ and $h=10^{-2}$ both approximations lead to very similar errors, but the computation time for 50 implicit Euler steps decreases from 6.22 seconds for the finite difference method to 0.93 seconds for our global spectral method. In this case the error is dominated by the implicit Euler scheme for both methods. In contrast, for $n=30$ and $h=10^{-4}$, the errors for the spectral method are smaller. For large τ the error of both approaches is dominated by the time discretization via the implicit Euler scheme. However, in the initial 750 time steps the error of the spatial discretization dominates for the finite difference scheme, whereas the global spectral method is able to represent u_τ accurately.



(a) Results for $n = 20$ and $h = 10^{-2}$.

(b) Results for $n = 20$ and $h = 10^{-4}$.

Figure 7.5 – Error plots for the parabolic PDE example in Section 7.4.3. We apply an implicit Euler scheme to compute approximations $u_\tau(x, y, z) \approx u(x, y, z, \tau h)$ and compare the solutions for the global spectral method (GSM) with solutions in $\mathbb{P}_{n,n,n}$ and a finite difference discretization (FD) on a regular $(n{+}1) \times (n{+}1) \times (n{+}1)$ grid. The absolute error $||u^*(\cdot, \cdot, \cdot, \tau h) - u_\tau(\cdot, \cdot, \cdot)||_\infty$ is estimated in each timestep from evaluations at 100 sample points. The relative error is computed from the estimation of the absolute error.

Remark 7.9. *Instead of using the implicit Euler scheme, we could generalize the ideas from [83] to derive a tensor-oriented exponential Euler scheme.*

Remark 7.10. *The methods presented in this work can also be used to solve two-dimensional parabolic PDEs on rectangles by treating time as third space variable in the discretization of the operator. We refer to [319] for more details on this approach.*

7.4.4 Eigenvalue problems

The methods presented in this work can be extended to solve PDE eigenvalue problems, in which we search eigenvalues λ and eigenfunctions u satisfying the equation $\mathcal{L}u = \lambda u$ complemented with homogeneous Dirichlet boundary conditions. We are particularly interested in finding the eigenvalue with minimal absolute value. For this purpose we employ the inverse iteration algorithm [137]. Starting from an initial function u_0 we iteratively compute an approximation of the eigenfunction. For $s = 1, 2, \dots,$ we compute u_s as solution of the PDE

$$\mathcal{L}u_s = \frac{u_{s-1}}{||u_{s-1}||_{\mathcal{L}^2}}. \tag{7.28}$$

We approximate the eigenvalue using the Rayleigh quotient $\frac{1}{\lambda} \approx \frac{\langle u_{s-1}, u_s \rangle}{\langle u_{s-1}, u_{s-1} \rangle}$, where $\langle \cdot, \cdot \rangle$ denotes the standard $\mathcal{L}^2$ scalar product. We again proceed by discretizing the differential operator and the function u to solve Equation (7.28) using the spectral method introduced in Section 7.3.

Let the functions $u, v \in \mathbb{P}_{n_1,n_2,n_3}$ be given in the form of (7.4). We evaluate the norm as $||u||^2_{\mathcal{L}^2} = \langle u, u\rangle$ and the scalar product $\langle u, v\rangle$ by interpolating the function uv using tensorized Chebyshev polynomial basis functions as in [100, 224]. In a second step, we integrate the approximation of uv using the exact values for the integrals of the basis functions [322, Theorem 19.2].

We test our method for the elliptic PDE eigenvalue problem

$$-\Delta u(x, y, z) + v(x, y, z)u(x, y, z) = \lambda u(x, y, z), \tag{7.29}$$

with potential $v(x, y, z) = \sin(\pi/2(x + 1)) \sin(\pi/2(y + 1)) \sin(\pi/2(z + 1))$ as in [154, 208]. Due to the potential $v(x, y, z)$ the inverse iteration steps (7.28) are Helmholtz equations (7.22) with non-constant separable coefficients. We discretize these by adding a discretization of $v(x, y, z)\mathcal{I}$ with $R = 1$ to a discretization of the Laplacian. The resulting discretized PDE (7.10) has $R = 4$ and we apply preconditioned GMRES with restarting using a Helmholtz equation with constant coefficient $||v||_{\mathcal{L}^2}$ to compute solutions as in Section 7.4.2. We compare our global spectral method to a finite difference scheme on a regular grid for solving (7.28). In Figure 7.6 we observe that the eigenvalues computed as Rayleigh coefficients converge at a faster rate for the global spectral method than the finite difference approach. Already for $n = 20$ the global spectral method reaches an error close to machine precision.

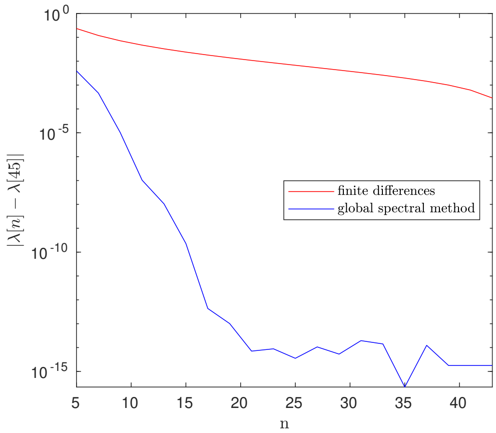


Figure 7.6 – Eigenvalue convergence rate for the PDE eigenvalue problem (7.29) computed as Rayleigh coefficients from u_{50} in the inverse iteration method, where the solutions u_s are computed via the global spectral method with $u_s \in \mathbb{P}_{n,n,n}$ and a finite difference method on a regular $(n+1) \times (n+1) \times (n+1)$ grid. We denote the eigenvalue obtained for n by $\lambda[n]$. Since the exact λ is not known, we plot $|\lambda[n] - \lambda[45]|$ for both algorithms, where $\lambda[45]$ is computed with the respective algorithm.

Remark 7.11. *We would like to point out that our approach can be used to compute a* *basis for the (continuous) Krylov subspace, which is used in the Arnoldi method [15, 137] and in the Least-squares spectral method in [162].*

8 Conclusions and outlook

In this thesis, we demonstrated that three very different high-dimensional problems can be solved efficiently by identifying and exploiting underlying low-rank tensor structures to overcome the curse of dimensionality.

Functional low-rank approximation In Chapters 3 and 4, we developed two novel algorithms to compute functional low-rank approximations of multivariate functions based on selected function evaluations. The key point in deriving these algorithms has been that low-rank approximations of the evaluation tensor for tensorized interpolation lead to particular types of functional low-rank approximations. Compared to previously existing algorithms, we managed to significantly reduce the number of function evaluations needed to obtain an approximation with prescribed accuracy for most functions.

Throughout this thesis, our approximation algorithms have been derived based on expansions in terms of tensorized polynomials. Extensions to other basis functions are possible and might offer certain advantages. For example, Ballfun [43] avoids the introduction of artificial boundaries when approximating functions on the ball by combining a mix of Chebyshev and Fourier basis functions with additional symmetry constraints. So far, we have focused on approximations in the functional Tucker or the (extended) functional TT format. While we have demonstrated that many functions can be well approximated in these formats, it might be possible to compress certain functions even more when using other functional low-rank approximation formats [21]. To obtain such approximations efficiently, we will need to develop specialized approximation algorithms. Those algorithms potentially could reuse parts of the algorithms presented in this work, such as first computing a Tucker approximation using Algorithm 9 before approximating the core tensor in the desired format. Future work could additionally cover how to compute numerically with functions [323] once they are compressed in functional low-rank formats. For arithmetic operations, one could perform certain operations by directly manipulating the low-rank approximation of the evaluation and the coefficient tensor. For instance, the multiplication of functions could be seen as Hadamard product of the corresponding low-rank approximations of the evaluation tensor [204].

In Chapter 7, we derived a global spectral method for solving three-dimensional linear PDEs on cubes with very high accuracy. Our numerical experiments demonstrated that using the blocked recursive solver [69] to either solve the resulting equations directly or as a preconditioner when the equations are not Laplace-like vastly outperforms all existing methods. We additionally presented the versatility of our method by the extension to eigenvalue and time-dependent problems.

In future work, additional features included in Chebop2 [319] such as adaptivity and input parsing could be incorporated into the current implementation of our global spectral method. Currently, the computational complexity of our approach is heavily influenced by the storage needed to store the full coefficient tensors $\mathcal{U}$, $\mathcal{F}$ and $\mathcal{G}_\ell$ for representing the solution, the right-hand side and the boundary conditions, respectively. Under the assumption that both the solution of the PDE and the right-hand side can be well approximated in functional low-rank formats, also these coefficient tensors can be well approximated in the corresponding low-rank tensor formats. Such approximations have the potential to drastically reduce the storage complexity. Exploiting this potential requires determining suitable approximation formats and using a specialized solver for the chosen formats. For instance, when choosing the Tucker or the TT format, we can use the ideas from [289] to solve Laplace-like equations directly in these low-rank formats. Alternatively, one could add the boundary conditions as penalty terms to the discretized PDE as in [162] and apply standard techniques such as Riemannian optimization [264] or alternating linear schemes [171] to solve the resulting system in the TT format. The computational complexity of time-dependent problems could be reduced using rank-adaptive dynamical low-rank approximations [62, 64, 90]. We also would like to point out that extending our global spectral method to higher dimensional PDEs on hypercubes is straightforward, but it might no longer be feasible to solve the resulting system numerically without exploiting additional low-rank structure of the coefficient tensors.

Multi-marginal optimal transport In Chapter 5, we analyzed the impact of approximating the Gibbs kernel on the Sinkhorn algorithm for solving entropically regularized multi-marginal optimal transport problems. We demonstrated that the introduction of low-rank approximations offers the potential to reduce the computational complexity of evaluating marginals by orders of magnitude. For transport plans defined via graphical models, we showed that constructing the low-rank approximation based on the dual graphical model can be faster and more accurate than the direct approximation of the Gibbs kernel. An application of our method is presented by the drastic reduction of the computation time for transferring colors from several images onto one target image.

So far, we computed the approximation $\tilde{\mathcal{K}}$ of the Gibbs kernel $\mathcal{K}$ using classical low-rank approximation algorithms, which are designed to minimize $\|\mathcal{K} - \tilde{\mathcal{K}}\|_\infty$ with respect to the uniform or the Frobenius norm. However, our error bounds indicate that we are actually interested in finding potentially very different approximations for which $\|\log(\mathcal{K}) - \log(\tilde{\mathcal{K}})\|_\infty$ is minimized. Efficient algorithms to compute such approximations are still subject to future work. The design of these algorithms can potentially be inspired by [172, 330], where approximations in the CP format are computed such that non-standard objective functions are minimized. In [112], the related problem of finding low-rank approximations of elementwise exponentials is studied in the context of log-normal random fields. For multi-marginal optimal transport problems arising in the context of density functional theory [35], several obstacles need to be overcome before our approach can be put into practice. The Coulomb cost leads to zero entries on the diagonal of the Gibbs kernel. Any suitable approximation of the Gibbs kernel would need to approximate these entries exactly, which is usually not feasible without either adding sparse correction terms to the low-rank approximation as for instance in [213, 275] or generalizing the idea of using hierarchical matrices [234] to tensors. Moreover, the underlying graphical model leads to a fully connected tensor network that can not be contracted efficiently.

Self-diffusion matrix In Chapter 6, we introduced a novel approach to compute the self-diffusion matrix of a tagged particle process. We exploited the fact that the latter quantity can be expressed via the solution of a high-dimensional deterministic minimization problem to develop an alternating optimization scheme to build a low-rank approximation of this solution. Our numerical experiments showed that the variance in the computed self-diffusion matrix is much smaller when using our novel approach compared to using standard sampling based methods with the same amount of computational resources.

There still remain several interesting open questions that need to be investigated in future work. As already described in Section 6.3.5, the current implementation of our low-rank algorithm is limited in terms of the size of the computational domain for which the algorithm can be run due to round-off errors. We still believe that our approach is promising since the error linked to the truncation of the computational domain is small in comparison to statistical errors. In this thesis, all our experiments have been concerned with two-dimensional tagged particle systems. Extensions to higher dimensional-systems are possible, but the rounding issues become more and more pressing. So far, we only derived a finite volume scheme to solve a simplified cross-diffusion system involving the computed self-diffusion matrices. It still remains to develop a convergent entropy diminishing finite volume scheme for hydrodynamic limits of multi-species exclusion processes.

Bibliography

[1] L. Agelas, R. Eymard, and R. Herbin. *A nine-point finite volume scheme for the simulation of diffusion in heterogeneous media.* C. R. Math. Acad. Sci. Paris 347.11-12 (2009), pp. 673–676.

[2] M. Agueh and G. Carlier. *Barycenters in the Wasserstein space.* SIAM J. Math. Anal. 43.2 (2011), pp. 904–924.

[3] S. Ahmadi-Asl et al. *Cross tensor approximation methods for compression and dimensionality reduction.* IEEE Access 9 (2021), pp. 150809–150838.

[4] M. Ali and A. Nouy. *Approximation with tensor networks. Part I: Approximation spaces.* arXiv e-prints (2020), arXiv:2007.00118.

[5] M. Ali and A. Nouy. *Approximation with tensor networks. Part II: Approximation rates for smoothness classes.* arXiv e-prints (2020), arXiv:2007.00128.

[6] M. Ali and A. Nouy. *Approximation with tensor networks. Part III: Multivariate approximation.* arXiv e-prints (2021), arXiv:2101.11932.

[7] J. M. Altschuler and E. Boix-Adsera. *Polynomial-time algorithms for multi-marginal optimal transport problems with structure.* Math. Program. (2022), pp. 1–72.

[8] J. M. Altschuler and E. Boix-Adserà. *Hardness results for multimarginal optimal transport problems.* Discrete Optim. 42 (2021), Paper No. 100669, 21.

[9] J. Altschuler, J. Weed, and P. Rigollet. *Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration.* In: Adv. Neural Inf. Process. Syst. 30. 2017, pp. 1964–1974.

[10] J. Altschuler et al. *Massively scalable Sinkhorn distances via the Nyström method.* In: Adv. Neural Inf. Process. Syst. 32. 2019, pp. 4427–4437.

[11] A. Ammar et al. *Proper generalized decomposition of time-multiscale models.* Internat. J. Numer. Methods Eng. 90.5 (2012), pp. 569–596.

[12] J. An and A. Owen. *Quasi-regression.* J. Complexity 17.4 (2001), pp. 588–607.

[13] C. Arita, P. L. Krapivsky, and K. Mallick. *Bulk diffusion in a kinetically constrained lattice gas.* J. Phys. A 51 (2018), p. 125002.

[14] C. Arita, P. L. Krapivsky, and K. Mallick. *Variational calculation of transport coefficients in diffusive lattice gases.* Phys. Rev. E 95 (2017), p. 032121.

[15] W. E. Arnoldi. *The principle of minimized iteration in the solution of the matrix eigenvalue problem.* Quart. Appl. Math. 9 (1951), pp. 17–29.

[16] J. L. Aurentz and L. N. Trefethen. *Chopping a Chebyshev series.* ACM Trans. Math. Software 43.4 (2017), pp. 1–21.

[17] F. A. Ba and M. Quellmalz. *Accelerating the Sinkhorn algorithm for sparse multi-marginal optimal transport via fast Fourier transforms.* Algorithms 15.9 (2022).

[18] I. Babuška and M. Suri. *The p- and h-p versions of the finite element method, an overview.* Comput. Methods Appl. Mech. Eng. 80 (1990), pp. 5–26.

[19] F. Bach. *Breaking the curse of dimensionality with convex neural networks.* J. Mach. Learn. Res. 18 (2017), pp. 1–53.

[20] M. Bachmayr and A. Cohen. *Kolmogorov widths and low-rank approximations of parametric elliptic PDEs.* Math. Comp. 86.304 (2017), pp. 701–724.

[21] M. Bachmayr, A. Nouy, and R. Schneider. *Approximation by tree tensor networks in high dimensions: Sobolev and compositional functions.* arXiv e-prints (2021), arXiv:2112.01474.

[22] M. Bachmayr, R. Schneider, and A. Uschmajew. *Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations.* Found. Comput. Math. 16.6 (2016), pp. 1423–1472.

[23] A. Bakhta and V. Ehrlacher. *Cross-diffusion systems with non-zero flux and moving boundary conditions.* ESAIM Math. Model. Numer. Anal. 52.4 (2018), pp. 1385–1415.

[24] J. Ballani and L. Grasedyck. *A projection method to solve linear systems in tensor format.* Numer. Linear Algebra Appl. 20.1 (2013), pp. 27–43.

[25] R. Ballester-Ripoll, E. G. Paredes, and R. Pajarola. *Sobol tensor trains for global sensitivity analysis.* Reliab. Eng. Syst. Saf. 183 (2019), pp. 311–322.

[26] R. Bapat. *D_1AD_2 theorems for multidimensional matrices.* Linear Algebra Appl. 48 (1982), pp. 437–442.

[27] Z. Battles and L. N. Trefethen. *An extension of MATLAB to continuous functions and operators.* SIAM J. Sci. Comput. 25.5 (2004), pp. 1743–1770.

[28] Y. Bazilevs et al. *3D simulation of wind turbine rotors at full scale. Part II: Fluid–structure interaction modeling with composite blades.* Int. J. Numer. Meth. Fluids 65.1-3 (2010), pp. 236–253.

[29] M. Bebendorf and S. Rjasanow. *Adaptive low-rank approximation of collocation matrices.* Computing 70.1 (2003), pp. 1–24.

[30] M. Bebendorf. *Approximation of boundary element matrices.* Numer. Math. 86.4 (2000), pp. 565–589.

[31] M. Bebendorf. *Hierarchical Matrices.* Vol. 63. Lect. Notes Comput. Sci. Eng. Springer, 2008.

[32] F. Beier et al. *Unbalanced multi-marginal optimal transport.* J. Math. Imaging Vis. (2021), pp. 1–20.

[33] R. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[34] J.-D. Benamou. *Optimal transportation, modelling and numerical simulation.* Acta Numer. 30 (2021), pp. 249–325.

[35] J.-D. Benamou, G. Carlier, and L. Nenna. *A numerical method to solve multi-marginal optimal transport problems with Coulomb cost.* In: Splitting Methods in Communication, Imaging, Science, and Engineering. Springer, 2016, pp. 577–601.

[36] J.-D. Benamou, G. Carlier, and L. Nenna. *Generalized incompressible flows, multi-marginal transport and Sinkhorn algorithm.* Numer. Math. 142.1 (2019), pp. 33–54.

[37] J.-D. Benamou et al. *Iterative Bregman projections for regularized transportation problems.* SIAM J. Sci. Comput. 37.2 (2015), A1111–A1138.

[38] J.-P. Berrut and L. N. Trefethen. *Barycentric Lagrange interpolation.* SIAM Rev. 46.3 (2004), pp. 501–517.

[39] G. Beylkin and M. J. Mohlenkamp. *Algorithms for numerical analysis in high dimensions.* SIAM J. Sci. Comput. 26.6 (2005), pp. 2133–2159.

[40] G. Beylkin and M. J. Mohlenkamp. *Numerical operator calculus in higher dimensions.* Proc. Natl. Acad. Sci. USA 99.16 (2002), pp. 10246–10251.

[41] D. Bigoni, A. P. Engsig-Karup, and Y. M. Marzouk. *Spectral tensor-train decomposition.* SIAM J. Sci. Comput. 38.4 (2016), A2405–A2439.

[42] O. Blondel and C. Toninelli. *Kinetically constrained lattice gases: Tagged particle diffusion.* Ann. Inst. Henri Poincaré Probab. Stat. 54.4 (2018), pp. 2335–2348.

[43] N. Boullé and A. Townsend. *Computing with functions in the ball.* SIAM J. Sci. Comput. 42.4 (2020), pp. C169–C191.

[44] N. Boumal. *An Introduction to Optimization on Smooth Manifolds.* Cambridge University Press, 2023.

[45] C. Boutsidis and D. P. Woodruff. *Optimal CUR matrix decompositions.* SIAM J. Comput. 46.2 (2017), pp. 543–589.

[46] J. P. Boyd and R. Petschek. *The relationships between Chebyshev, Legendre and Jacobi polynomials: The generic superiority of Chebyshev polynomials and three important exceptions.* J. Sci. Comput. 59.1 (2014), pp. 1–27.

[47] D. Braess and W. Hackbusch. *Approximation of* $1/x$ *by exponential sums in* $[1, \infty)$. IMA J. Numer. Anal. 25.4 (2005), pp. 685–697.

[48] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods.* 3rd. Vol. 15. Texts Appl. Math. Springer, 2008.

[49] M. Bruna and S. J. Chapman. *Diffusion of multiple species with excluded-volume effects.* J. Chem. Phys. 137.20 (2012), p. 204116.

[50] H.-J. Bungartz and M. Griebel. *Sparse grids.* Acta Numer. 13 (2004), pp. 147–269.

[51] M. Burger et al. *Nonlinear cross-diffusion with size exclusion.* SIAM J. Math. Anal. 42.6 (2010), pp. 2842–2871.

[52] H. Cai et al. *Fast robust tensor principal component analysis via fiber CUR decomposition.* In: IEEE/CVF Int. Conf. Comput. Vis. Workshop. 2021, pp. 189–197.

[53] H. Cai et al. *Mode-wise tensor decompositions: Multi-dimensional generalizations of CUR decompositions.* J. Mach. Learn. Res. 22 (2021), Paper No. 185, 36.

[54] C. F. Caiafa and A. Cichocki. *Generalizing the column-row matrix decomposition to multi-way arrays.* Linear Algebra Appl. 433.3 (2010), pp. 557–573.

[55] E. J. Candès and T. Tao. *The power of convex relaxation: Near-optimal matrix completion.* IEEE Trans. Inform. Theory 56.5 (2010), pp. 2053–2080.

[56] L. F. Canino et al. *Numerical solution of the Helmholtz equation in 2D and 3D using a high-order Nyström discretization.* J. Comput. Phys. 146.2 (1998), pp. 627–663.

[57] J. Cao et al. *Multi-marginal wasserstein GAN.* In: Adv. Neural Inf. Process. Syst. 32. 2019, pp. 1776–1786.

[58] G. Carlier. *On the linear convergence of the multimarginal Sinkhorn algorithm.* SIAM J. Optim. 32.2 (2022), pp. 786–794.

[59] G. Carlier, A. Oberman, and E. Oudet. *Numerical methods for matching for teams and Wasserstein barycenters.* ESAIM Math. Model. Numer. Anal. 49.6 (2015), pp. 1621–1642.

[60] J. D. Carroll and J.-J. Chang. *Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition.* Psychometrika 35.3 (1970), pp. 283–319.

[61] J. D. Carroll, S. Pruzansky, and J. B. Kruskal. *CANDELINC: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters.* Psychometrika 45.1 (1980), pp. 3–24.

[62] G. Ceruti, J. Kusch, and C. Lubich. *A rank-adaptive robust integrator for dynamical low-rank approximation.* BIT 62.4 (2022), pp. 1149–1174.

[63] G. Ceruti and C. Lubich. *An unconventional robust integrator for dynamical low-rank approximation.* BIT 62.1 (2022), pp. 23–44.

[64] G. Ceruti, C. Lubich, and D. Sulz. *Rank-adaptive time integration of tree tensor networks.* arXiv e-prints (2022), arXiv:2201.10291.

[65] S. Chaturantabut and D. C. Sorensen. *Discrete empirical interpolation for nonlinear model reduction.* In: 48h IEEE Conf. Decis. Control and 28th Chin. Control Conf. CCC. 2009, pp. 4316–4321.

[66] S. Chaturantabut and D. C. Sorensen. *Nonlinear model reduction via discrete empirical interpolation.* SIAM J. Sci. Comput. 32.5 (2010), pp. 2737–2764.

[67] P. L. Chebyshev. *Théorie des Mécanismes Connus sous le Nom de Parallélogrammes.* Imprimerie de l'Académie impériale des sciences, 1853.

[68] D. Chen et al. *Limit theorems for the tagged particle in exclusion processes on regular trees.* Electron. Commun. Probab. 24 (2019), Paper No. 2, 10.

[69] M. Chen and D. Kressner. *Recursive blocked algorithms for linear systems with Kronecker product structure.* Numer. Algorithms 84.3 (2020), pp. 1199–1216.

[70] Z. Chen and L. Lu. *A projection method and Kronecker product preconditioner for solving Sylvester tensor equations.* Sci. China Math. 55.6 (2012), pp. 1281–1292.

[71] A. Chertkov, G. Ryzhakov, and I. Oseledets. *Black box approximation in the tensor train format initialized by ANOVA decomposition.* arXiv e-prints (2022), arXiv:2208.03380.

[72] A. Cichocki et al. *Tensor decompositions for signal processing applications: From two-way to multiway component analysis.* IEEE Signal Process. Mag. 32.2 (2015), pp. 145–163.

[73] A. Cichocki et al. *Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions.* Found. Trends Mach. Learn. 9.4–5 (2016), pp. 249–429.

[74] C. W. Clenshaw and A. R. Curtis. *A method for numerical integration on an automatic computer.* Numer. Math. 2 (1960), pp. 197–205.

[75] A. Cohen, R. Devore, and C. Schwab. *Analytic regularity and polynomial approximation of parametric and stochastic elliptic PDEs.* Anal. Appl. (Singap.) 9.1 (2011), pp. 11–47.

[76] P. G. Constantine. *Active Subspaces.* Vol. 2. SIAM Spotlights. SIAM, 2015.

[77] P. G. Constantine, C. Kent, and T. Bui-Thanh. *Accelerating Markov chain Monte Carlo with active subspaces.* SIAM J. Sci. Comput. 38.5 (2016), A2779–A2805.

[78] A. Cortinovis. *Fast deterministic and randomized algorithms for low-rank approximation, matrix functions, and trace estimation.* PhD thesis. EPFL, 2022.

[79] A. Cortinovis and D. Kressner. *Low-rank approximation in the Frobenius norm by column and row subset selection.* SIAM J. Matrix Anal. Appl. 41.4 (2020), pp. 1651–1673.

[80] A. Cortinovis, D. Kressner, and S. Massei. *On maximum volume submatrices and cross approximation for symmetric semidefinite and diagonally dominant matrices.* Linear Algebra Appl. 593 (2020), pp. 251–268.

[81] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* 2nd. Wiley-Interscience, 2006.

[82] M. Cuturi. *Sinkhorn distances: Lightspeed computation of optimal transport.* In: Adv. Neural Inf. Process. Syst. 26. 2013, pp. 2292–2300.

[83] M. C. D'Autilia, I. Sgura, and V. Simoncini. *Matrix-oriented discretization methods for reaction-diffusion PDEs: Comparisons and applications.* Comput. Math. Appl. 79.7 (2020), pp. 2067–2085.

[84] J. Dabaghi, V. Ehrlacher, and C. Strössner. *Computation of the self-diffusion coefficient with low-rank tensor methods: Application to the simulation of a cross-diffusion system.* arXiv e-prints (2021), 2111.11349. To appear in ESAIM Proc. Surveys.

[85] J. Dabaghi, V. Ehrlacher, and C. Strössner. *Tensor approximation of the self-diffusion matrix of tagged particle processes.* arXiv e-prints (2022), arXiv:2204.03943. To appear in J. Comput. Phys..

[86] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration.* 2nd. Comput. Sci. Appl. Math. Academic Press, 1984.

[87] L. De Lathauwer. *Decompositions of a higher-order tensor in block terms. II. Definitions and uniqueness.* SIAM J. Matrix Anal. Appl. 30.3 (2008), pp. 1033–1066.

[88] L. De Lathauwer, B. De Moor, and J. Vandewalle. *A multilinear singular value decomposition.* SIAM J. Matrix Anal. Appl. 21.4 (2000), pp. 1253–1278.

[89] A. De Masi and E. Presutti. *Mathematical Methods for Hydrodynamic Limits.* Vol. 1501. Lecture Notes in Math. Springer, 1991.

[90] A. Dektor, A. Rodgers, and D. Venturi. *Rank-adaptive tensor methods for high-dimensional nonlinear PDEs.* J. Sci. Comput. 88:36.36 (2021).

[91] A. Dektor and D. Venturi. *Dynamically orthogonal tensor methods for high-dimensional nonlinear PDEs.* J. Comput. Phys. 404 (2020), p. 103501.

[92] A. Dektor and D. Venturi. *Tensor rank reduction via coordinate flows.* arXiv e-prints (2022), arXiv:2207.11955.

[93] A. Deshpande and L. Rademacher. *Efficient volume sampling for row/column subset selection.* In: 51st Annu. IEEE Symp. Found. Comput. Sci. FOCS. 2010, pp. 329–338.

[94] H. Dette and A. Pepelyshev. *Generalized Latin hypercube design for computer experiments.* Technometrics 52.4 (2010), pp. 421–429.

[95] H. P. Deutsch and K. Binder. *Interdiffusion and self-diffusion in polymer mixtures: A Monte Carlo study.* J. Chem. Phys. 94.3 (1991), pp. 2294–2304.

[96] S. Di Marino and A. Gerolin. *An optimal transport approach for the Schrödinger bridge problem and convergence of Sinkhorn algorithm.* J. Sci. Comput. 85.2 (2020), pp. 1–28.

[97] S. Di Marino, A. Gerolin, and L. Nenna. *Optimal transportation theory with repulsive costs.* In: Topological Optimization and Optimal Transport. Vol. 17. Radon Ser. Comput. Appl. Math. De Gruyter, 2017, pp. 204–256.

[98] J. Dieterich and B. Hartke. *Empirical review of standard benchmark functions using evolutionary global optimization.* Applied Math. 3.10 (2012).

[99] S. Dolgov and B. Khoromskij. *Two-level QTT-Tucker format for optimized tensor calculus.* SIAM J. Matrix Anal. Appl. 34.2 (2013), pp. 593–623.

[100] S. Dolgov, D. Kressner, and C. Strössner. *Functional Tucker approximation using Chebyshev interpolation.* SIAM J. Sci. Comput. 43.3 (2021), A2190–A2210.

[101] S. Dolgov and R. Scheichl. *A hybrid alternating least squares-TT-cross algorithm for parametric PDEs.* SIAM/ASA J. Uncertain. Quantif. 7.1 (2019), pp. 260–291.

[102] T. A. Driscoll, F. Bornemann, and L. N. Trefethen. *The chebop system for automatic solution of differential equations.* BIT 48.4 (2008), pp. 701–723.

[103] T. A. Driscoll and N. Hale. *Rectangular spectral collocation.* IMA J. Numer. Anal. 36.1 (2016), pp. 108–132.

[104] T. A. Driscoll, N. Hale, and L. N. Trefethen. *Chebfun Guide.* Pafnuty Publications, 2014.

[105] Z. Drmač and A. K. Saibaba. *The discrete empirical interpolation method: canonical structure and formulation in weighted inner product spaces.* SIAM J. Matrix Anal. Appl. 39.3 (2018), pp. 1152–1180.

[106] P. Dvurechensky, A. Gasnikov, and A. Kroshnin. *Computational optimal transport: Complexity by accelerated gradient descent is better than by Sinkhorn's algorithm.* In: 35th Int. Conf. Mach. Learn. 2018, pp. 1367–1376.

[107] D. Ebling et al. *Multiphysics simulation of thermoelectric systems for comparison with experimental device performance.* J. Electron. Mater. 38.7 (2009), pp. 1456–1461.

[108] M. Eigel, R. Gruhlke, and M. Marschall. *Low-rank tensor reconstruction of concentrated densities with application to Bayesian inversion.* Stat. Comput. 32.2 (2022), Paper No. 27, 27.

[109] L. Einkemmer and C. Lubich. *A low-rank projector-splitting integrator for the Vlasov-Poisson equation.* SIAM J. Sci. Comput. 40.5 (2018), B1330–B1360.

[110] F. Elvander et al. *Multi-marginal optimal transport using partial information with applications in robust localization and sensor fusion.* Signal Process. 171 (2020), p. 107474.

[111] C. Erignoux. *Limite hydrodynamique pour un gaz sur réseau de particules actives.* PhD thesis. Université Paris Diderot, 2016.

[112] M. Espig et al. *Efficient low-rank approximation of the stochastic Galerkin matrix in tensor formats.* Comput. Math. Appl. 67.4 (2014), pp. 818–829.

[113] R. Eymard, T. Gallouët, and R. Herbin. *A finite volume scheme for anisotropic diffusion problems.* C. R. Math. Acad. Sci. Paris 339.4 (2004), pp. 299–302.

[114] R. Eymard, T. Gallouët, and R. Herbin. *A new finite volume scheme for anisotropic diffusion problems on general grids: Convergence analysis.* C. R. Math. Acad. Sci. Paris 344.6 (2007), pp. 403–406.

[115] R. Eymard, T. Gallouët, and R. Herbin. *Convergence of finite volume schemes for semilinear convection diffusion equations.* Numer. Math. 82.1 (1999), pp. 91–116.

[116] R. Eymard, T. Gallouët, and R. Herbin. *Finite volume methods.* In: Handbook of Numerical Analysis, VII. North-Holland, 2000, pp. 713–1020.

[117] A. Falcó, W. Hackbusch, and A. Nouy. *Tree-based tensor formats.* SeMA Journal (2018).

[118] J. Fan et al. *On the complexity of the optimal transport problem with graph-structured cost.* In: 25th Int. Conf. Artif. Intell. Stat. 2022, pp. 9147–9165.

[119] G. Favier and A. L. de Almeida. *Overview of constrained PARAFAC models.* EURASIP J. Adv. Signal Process. 142 (2014), pp. 1–25.

[120] R. Ferrando and E. Scalas. *Self-diffusion in a 2D lattice gas with lateral interactions.* Surf. Sci. 281.1-2 (1993), pp. 178–190.

[121] J. Feydy et al. *Optimal transport for diffeomorphic registration.* In: MICCAI. 2017, pp. 291–299.

[122] B. Fornberg. *A Practical Guide to Pseudospectral Methods.* Vol. 1. Cambridge Monogr. Appl. Comput. Math. Cambridge University Press, 1996.

[123] A. I. J. Forrester, A. Sóbester, and A. J. Keane. *Engineering Design via Surrogate Modelling: A Practical Guide.* John Wiley & Sons, 2008.

[124] D. Fortunato, N. Hale, and A. Townsend. *The ultraspherical spectral element method.* J. Comput. Phys. 436 (2021), p. 110087.

[125] D. Fortunato and A. Townsend. *Fast Poisson solvers for spectral methods.* IMA J. Numer. Anal. 40.3 (2020), pp. 1994–2018.

[126] J. Franklin and J. Lorenz. *On the scaling of multidimensional matrices.* Linear Algebra Appl. 114/115 (1989), pp. 717–735.

[127] S. Friedland. *Tensor optimal transport, distance between sets of measures and tensor scaling.* arXiv e-prints (2020), arXiv:2005.00945.

[128] J. H. Friedman. *Multivariate adaptive regression splines.* Ann. Statist. 19.1 (1991), pp. 1–141.

[129] G. Friesecke and M. Penka. *The GenCol algorithm for high-dimensional optimal transport: General formulation and application to barycenters and Wasserstein splines.* arXiv e-prints (2022), arXiv:2209.09081.

[130] G. Friesecke, A. S. Schulz, and D. Vögler. *Genetic column generation: Fast computation of high-dimensional multimarginal optimal transport problems.* SIAM J. Sci. Comput. 44.3 (2022), A1632–A1654.

[131] X. Fu et al. *Computing large-scale matrix and tensor decomposition with structured factors: A unified nonconvex optimization perspective.* IEEE Signal Process. Mag. 37.5 (2020), pp. 78–94.

[132] N. Gantert and D. Schmid. *The speed of the tagged particle in the exclusion process on Galton-Watson trees.* Electron. J. Probab. 25 (2020), Paper No. 71, 27.

[133] S. Garreis and M. Ulbrich. *Constrained optimization with low-rank tensors and applications to parametric problems with PDEs.* SIAM J. Sci. Comput. 39.1 (2017), A25–A54.

[134] A. Genevay, G. Peyre, and M. Cuturi. *Learning generative models with Sinkhorn divergences.* In: Int. Conf. Artif. Intell. Stat. 2018, pp. 1608–1617.

[135] W. M. Gentleman. *Algorithm 424: Clenshaw-Curtis quadrature [D1].* Commun. ACM 15.5 (1972), pp. 353–355.

[136] A. Genz. *A package for testing multiple integration subroutines.* In: Numerical Integration. NATO ASI Series. Springer, 1987, pp. 337–340.

[137] G. H. Golub and C. F. Van Loan. *Matrix Computations.* 4th. Johns Hopkins Stud. Math. Sci. Johns Hopkins University Press, 2013.

[138] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. *A theory of pseudoskeleton approximations.* Linear Algebra Appl. 261 (1997), pp. 1–21.

[139] A. Gorodetsky. *Continuous low-rank tensor decompositions, with applications to stochastic optimal control and data assimilation.* PhD thesis. MIT, 2017.

[140] A. A. Gorodetsky and J. D. Jakeman. *Gradient-based optimization for regression in the functional tensor-train format.* J. Comput. Phys. 374 (2018), pp. 1219–1238.

[141] A. Gorodetsky, S. Karaman, and Y. Marzouk. *A continuous analogue of the tensor-train decomposition.* Comput. Methods Appl. Mech. Eng. 347 (2019), pp. 59–84.

[142] A. Gorodetsky, S. Karaman, and Y. Marzouk. *High-dimensional stochastic optimal control using continuous tensor decompositions.* Int. J. Robot. Res. 37.2-3 (2018), pp. 340–377.

[143] D. Gottlieb and S. A. Orszag. *Numerical Analysis of Spectral Methods: Theory and Applications.* SIAM, 1977.

[144] R. B. Gramacy and H. K. H. Lee. *Adaptive design and analysis of supercomputer experiments.* Technometrics 51.2 (2009), pp. 130–145.

[145] L. Grasedyck, D. Kressner, and C. Tobler. *A literature survey of low-rank tensor approximation techniques.* GAMM-Mitt. 36.1 (2013), pp. 53–78.

[146] M. Griebel and H. Harbrecht. *Analysis of tensor approximation schemes for continuous functions.* Found. Comput. Math. (2021), pp. 1–22.

[147] M. Griebel and H. Harbrecht. *Approximation of bi-variate functions: Singular value decomposition versus sparse grids.* IMA J. Numer. Anal. 34.1 (2014), pp. 28–54.

[148] M. Griebel, H. Harbrecht, and R. Schneider. *Low-rank approximation of continuous functions in Sobolev spaces with dominating mixed smoothness.* arXiv e-prints (2022), arXiv:2203.04100.

[149] M. Gu and S. C. Eisenstat. *Efficient algorithms for computing a strong rank-revealing QR factorization.* SIAM J. Sci. Comput. 17.4 (1996), pp. 848–869.

[150] I. Haasler et al. *Multi-marginal optimal transport and probabilistic graphical models.* IEEE Trans. Inf. Theory 67.7 (2021), pp. 4647–4668.

[151] I. Haasler et al. *Multimarginal optimal transport with a tree-structured cost and the Schrödinger bridge problem.* SIAM J. Control Optim. 59.4 (2021), pp. 2428–2453.

[152] W. Hackbusch and S. Kühn. *A new scheme for the tensor representation.* J. Fourier Anal. Appl. 15.5 (2009), pp. 706–722.

[153] W. Hackbusch. *Computation of best L^∞ exponential sums for $1/x$ by Remez' algorithm.* Comput. Vis. Sci. 20.1-2 (2019), pp. 1–11.

[154] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus.* Vol. 42. Springer Ser. Comput. Math. Springer, 2012.

[155] W. Hackbusch, L. Grasedyck, and S. Börm. *An introduction to hierarchical matrices.* In: EQUADIFF 10. Vol. 127. 2. 2002, pp. 229–241.

[156] N. Halko, P. G. Martinsson, and J. A. Tropp. *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.* SIAM Rev. 53.2 (2011), pp. 217–288.

[157] K. Hamm. *Generalized pseudoskeleton decompositions.* arXiv e-prints (2022), arXiv:2206.14905.

[158] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods.* Monogr. Statist. Appl. Probab. Springer, 1964.

[159] S. Hao and D. S. Sholl. *Self-diffusion and macroscopic diffusion of hydrogen in amorphous metals from first-principles calculations.* J. Chem. Phys. 130.24 (2009), p. 244705.

[160] S. Hao and P.-G. Martinsson. *A direct solver for elliptic PDEs in three dimensions based on hierarchical merging of Poincaré-Steklov operators.* J. Comput. Appl. Math. 308 (2016), pp. 419–434.

[161] R. A. Harshman. *Foundations of the PARAFAC Procedure: Models and Conditions for an Explanatory Multimodal factor Analysis.* Working Papers in Phonetics 16. University of California at Los Angeles, 1970.

[162] B. Hashemi and Y. Nakatsukasa. *Least-squares spectral methods for ODE eigenvalue problems.* SIAM J. Sci. Comput. 44.5 (2022), A3244–A3264.

[163] B. Hashemi and L. N. Trefethen. *Chebfun in three dimensions.* SIAM J. Sci. Comput. 39.5 (2017), pp. C341–C363.

[164] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral Methods for Time-Dependent Problems.* Vol. 21. Cambridge Monogr. Appl. Comput. Math. Cambridge University Press, 2007.

[165] N. J. Higham. *The numerical stability of barycentric Lagrange interpolation.* IMA J. Numer. Anal. 24.4 (2004), pp. 547–556.

[166] T. Hillen and K. J. Painter. *A user's guide to PDE models for chemotaxis.* J. Math. Biol. 58.1-2 (2009), pp. 183–217.

[167] G. E. Hinton and R. Zemel. *Autoencoders, minimum description length and Helmholtz free energy.* In: Adv. Neural Inf. Process. Syst. 6. Vol. 6. Morgan-Kaufmann, 1993.

[168] F. L. Hitchcock. *Multiple invariants and generalized rank of a p-way matrix or tensor.* J. Math. Phys. 7.1-4 (1928), pp. 39–79.

[169] S.-W. Ho and R. W. Yeung. *The interplay between entropy and variational distance.* IEEE Trans. Inform. Theory 56.12 (2010), pp. 5906–5929.

[170] S. Holtz, T. Rohwedder, and R. Schneider. *On manifolds of tensors of fixed TT-rank.* Numer. Math. 120.4 (2012), pp. 701–731.

[171] S. Holtz, T. Rohwedder, and R. Schneider. *The alternating linear scheme for tensor optimization in the tensor train format.* SIAM J. Sci. Comput. 34.2 (2012), A683–A713.

[172] D. Hong, T. G. Kolda, and J. A. Duersch. *Generalized canonical polyadic tensor decomposition.* SIAM Rev. 62.1 (2020), pp. 133–163.

[173] C. Huang and A. Darwiche. *Inference in belief networks: A procedural guide.* Internat. J. Approx. Reason. 15.3 (1996), pp. 225–263.

[174] C. Huang et al. *Efficient parallelization of tensor network contraction for simulating quantum computation.* Nat. Comput. Sci. 1.9 (2021), pp. 578–587.

[175] H. Igel. *Wave propagation in three-dimensional spherical sections by the Chebyshev spectral method.* Geophys. J. Int. 136.3 (1999), pp. 559–566.

[176] T. L. Jackson and H. M. Byrne. *A mechanical model of tumor encapsulation and transcapsular spread.* Math. Biosci. 180 (2002), pp. 307–328.

[177] M. Jamil and X.-S. Yang. *A literature survey of benchmark functions for global optimisation problems.* Int. J. Math. Model. Numer. Optim. 4.2 (2013), pp. 150–194.

[178] M. Jara. *Finite-dimensional approximation for the diffusion coefficient in the simple exclusion process.* Ann. Probab. 34.6 (2006), pp. 2365–2381.

[179] Y. Ji et al. *A survey on tensor techniques and applications in machine learning.* IEEE Access 7 (2019), pp. 162950–162990.

[180] T.-X. Jiang et al. *Nonnegative low rank tensor approximation and its application to multi-dimensional images.* Numer. Math. (2022), pp. 1–30.

[181] Y. W. Jonathan S Yedidia William Freeman. *Generalized belief propagation.* In: Adv. Neural Inf. Process. Syst. 13. 2000, pp. 689–695.

[182] A. Jüngel. *Entropy Methods for Diffusive Partial Differential Equations.* Springer Briefs Math. Springer, 2016.

[183] A. Jüngel. *The boundedness-by-entropy method for cross-diffusion systems.* Nonlinearity 28.6 (2015), pp. 1963–2001.

[184] G. E. Karniadakis and S. J. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics.* 2nd. Numer. Math. Sci. Comput. Oxford University Press, 2005.

[185] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations.* Vol. 16. Front. Appl. Math. SIAM, 1995.

[186] B. N. Khoromskij. *Structured rank-$(R_1, \dots, R_D)$ decomposition of function-related tensors in $\mathbb{R}^D$.* Comput. Methods Appl. Math. 6.2 (2006), pp. 194–220.

[187] B. N. Khoromskij. *$O(d \log N)$-quantics approximation of N-d tensors in high-dimensional numerical modeling.* Constr. Approx. 34.2 (2011), pp. 257–280.

[188] B. N. Khoromskij. *Tensor numerical methods for multidimensional PDEs: Theoretical analysis and initial applications.* ESAIM Proc. Surveys 48 (2015), pp. 1–28.

[189] B. N. Khoromskij. *Tensor Numerical Methods in Scientific Computing.* Vol. 19. Radon Ser. Comput. Appl. Math. De Gruyter, 2018.

[190] H. A. L. Kiers. *Towards a standardized notation and terminology in multiway analysis.* J. Chemom. 14.3 (2000), pp. 105–122.

[191] Y. Kim and S. Choi. *Nonnegative Tucker decomposition.* In: IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2007, pp. 1–8.

[192] C. Kipnis and S. R. S. Varadhan. *Central limit theorem for additive functionals of reversible Markov processes and applications to simple exclusions.* Comm. Math. Phys. 104.1 (1986), pp. 1–19.

[193] C. Kipnis, C. Landim, and S. Olla. *Hydrodynamical limit for a nongradient system: The generalized symmetric exclusion process.* Commun. Pure Appl. Math. 47.11 (1994), pp. 1475–1545.

[194] N. Kishore Kumar and J. Schneider. *Literature survey on low rank approximation of matrices.* Linear Multilinear Algebra 65.11 (2017), pp. 2212–2244.

[195] A. Klimke. *Sparse grid interpolation toolbox v5.1.1.* 2008.

[196] O. Koch and C. Lubich. *Dynamical tensor approximation.* SIAM J. Matrix Anal. Appl. 31.5 (2010), pp. 2360–2375.

[197] W. Koepf. *Hypergeometric Summation.* Adv. Lect. Math. Friedr. Vieweg & Sohn, 1998.

[198] T. G. Kolda and B. W. Bader. *Tensor decompositions and applications.* SIAM Rev. 51.3 (2009), pp. 455–500.

[199] S. Kolouri et al. *Optimal mass transport: Signal processing and machine-learning applications.* IEEE Signal Process. Mag. 34.4 (2017), pp. 43–59.

[200] T. Komorowski, C. Landim, and S. Olla. *Fluctuations in Markov Processes.* Vol. 345. Grundlehren Math. Wiss. Springer, 2012.

[201] K. Konakli and B. Sudret. *Global sensitivity analysis using low-rank tensor approximations.* Reliab. Eng. Syst. Saf. 156 (2016), pp. 64–83.

[202] K. Konakli and B. Sudret. *Polynomial meta-models with canonical low-rank approximations: Numerical insights and comparison to sparse polynomial chaos expansions.* J. Comput. Phys. 321 (2016), pp. 1144–1169.

[203] K. Konakli and B. Sudret. *Uncertainty quantification in high-dimensional spaces with low-rank tensor approximations.* In: 1st Int. Conf. Uncertain. Quantif. Comput. Sci. Eng. 2015.

[204] D. Kressner and L. Periša. *Recompression of Hadamard products of tensors in Tucker format.* SIAM J. Sci. Comput. 39.5 (2017), A1879–A1902.

[205] D. Kressner, M. Steinlechner, and B. Vandereycken. *Low-rank tensor completion by Riemannian optimization.* BIT 54.2 (2014), pp. 447–468.

[206] D. Kressner and C. Tobler. *Krylov subspace methods for linear systems with tensor product structure.* SIAM J. Matrix Anal. Appl. 31.4 (2009), pp. 1688–1714.

[207] D. Kressner and C. Tobler. *Low-rank tensor Krylov subspace methods for parametrized linear systems.* SIAM J. Matrix Anal. Appl. 32.4 (2011), pp. 1288–1316.

[208] D. Kressner and C. Tobler. *Preconditioned low-rank methods for high-dimensional elliptic PDE eigenvalue problems.* Comput. Methods Appl. Math. 11.3 (2011), pp. 363–381.

[209] L. Kronecker. *Über einige Interpolationsformeln für ganze Funktionen mehrerer Variablen.* In: Monatsberichte der Königlichen Preuß. Akademie der Wissenschaften zu Berlin aus dem Jahre 1865. Buchdruckerei der Königl. Akademie der Wissenschaften, 1866, pp. 686–691.

[210] C. Landim, S. Olla, and S. R. S. Varadhan. *Finite-dimensional approximation of the self-diffusion coefficient for the exclusion process.* Ann. Probab. 30.2 (2002), pp. 483–508.

[211] C. Léonard. *From the Schrödinger problem to the Monge-Kantorovich problem.* J. Funct. Anal. 262.4 (2012), pp. 1879–1920.

[212] K.-C. Li. *Sliced inverse regression for dimension reduction.* J. Amer. Statist. Assoc. 86.414 (1991), pp. 316–342.

[213] S. Li et al. *Low-rank tensor decomposition based anomaly detection for hyperspectral imagery.* In: IEEE Int. Conf. Image Process. (ICIP). 2015, pp. 4525–4529.

[214] T. M. Liggett. *Interacting Particle Systems.* Vol. 276. Grundlehren Math. Wiss. Springer, 1985.

[215] T. M. Liggett. *Stochastic Interacting Systems: Contact, Voter and Exclusion Processes.* Vol. 324. Grundlehren Math. Wiss. Springer, 1999.

[216] T. Lin, N. Ho, and M. I. Jordan. *On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms.* In: 36th Int. Conf. Mach. Learn. 2019, pp. 3982–3991.

[217] T. Lin et al. *On the complexity of approximating multimarginal optimal transport.* J. Mach. Learn. Res. 23.65 (2022), pp. 1–43.

[218] T.-Y. Lin et al. *Microsoft COCO: Common objects in context.* In: ECCV. 2014, pp. 740–755.

[219] G. Liu et al. *Module-based multiscale simulation of angiogenesis in skeletal muscle.* Theor. Biol. Med. Model. 8.6 (2011), pp. 1–21.

[220] G. J. Lord, C. E. Powell, and T. Shardlow. *An Introduction to Computational Stochastic PDEs.* Cambridge Texts Appl. Math. Cambridge University Press, 2014.

[221] T. H. Luu et al. *A new method for reconstruction of cross-sections using Tucker decomposition.* J. Comput. Phys. 345 (2017), pp. 189–206.

[222] P.-G. Martinsson and J. A. Tropp. *Randomized numerical linear algebra: Foundations and algorithms.* Acta Numer. 29 (2020), pp. 403–572.

[223] J. C. Mason. *Near-best multivariate approximation by Fourier series, Chebyshev series and Chebyshev interpolation.* J. Approx. Theory 28.4 (1980), pp. 349–358.

[224] J. C. Mason and D. C. Handscomb. *Chebyshev Polynomials*. Chapman and Hall/CRC, 2002.

[225] D. M. Mattox. *Handbook of Physical Vapor Deposition (PVD) Processing*. 2nd. William Andrew Publishing, 2010.

[226] D. McBride et al. *Computational modelling of variably saturated flow in porous media with complex three-dimensional geometries*. Internat. J. Numer. Methods Fluids 50.9 (2006), pp. 1085–1117.

[227] N. Metropolis and S. Ulam. *The Monte Carlo method*. J. Amer. Statist. Assoc. 44 (1949), pp. 335–341.

[228] R. Minster, Z. Li, and G. Ballard. *Parallel randomized Tucker decomposition algorithms*. arXiv e-prints (2022), arXiv:2211.13028.

[229] R. Minster, A. K. Saibaba, and M. E. Kilmer. *Efficient tensor-based approximations to kernel interactions*. SIAM-ALA presentation. 2021.

[230] R. Minster, A. K. Saibaba, and M. E. Kilmer. *Randomized algorithms for low-rank tensor decompositions in the Tucker format*. SIAM J. Math. Data Sci. 2.1 (2020), pp. 189–215.

[231] L. Mirsky. *Symmetric gauge functions and unitarily invariant norms*. Quart. J. Math. Oxford Ser. (2) 11 (1960), pp. 50–59.

[232] M. Moakher and P. G. Batchelor. *Symmetric positive-definite matrices: From geometry to applications and visualization*. In: Visualization and Processing of Tensor Fields. Math. Vis. Springer, 2006, pp. 285–298, 452.

[233] H. Moon, A. M. Dean, and T. J. Santner. *Two-stage sensitivity-based group screening in computer experiments*. Technometrics 54.4 (2012), pp. 376–387.

[234] M. Motamed. *A hierarchically low-rank optimal transport dissimilarity measure for structured data*. BIT (2022).

[235] M. Mozaffari et al. *Optimal transport theory for power-efficient deployment of unmanned aerial vehicles*. In: IEEE Int. Conf. Commun. 2016, pp. 1–6.

[236] E. A. Muravleva and I. V. Oseledets. *Approximate solution of linear systems with Laplace-like operators via cross approximation in the frequency domain*. Comput. Methods Appl. Math. 19.1 (2019), pp. 137–145.

[237] S. Narumi. *Some formulas in the theory of interpolation of many independent variables*. Tohoku Math. J. (1) 18 (1920), pp. 309–321.

[238] L. Nenna. *Numerical methods for multi-marginal optimal transportation*. PhD thesis. PSL Research University, 2016.

[239] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Vol. 63. CBMS-NSF Regional Conf. Ser. in Appl. Math. SIAM, 1992.

[240] *NIST Handbook of Mathematical Functions*. Cambridge University Press, 2010.

[241] A. Nouy. *Higher-order principal component analysis for the approximation of tensors in tree-based low-rank formats.* Numer. Math. 141.3 (2019), pp. 743–789.

[242] A. Nouy. *Low-rank methods for high-dimensional approximation and model order reduction.* In: Model Reduction and Approximation. Vol. 15. Comput. Sci. Eng. SIAM, 2017, pp. 171–226.

[243] A. Nouy. *Low-rank tensor methods for model order reduction.* In: Handbook of Uncertainty Quantification. Springer, 2017, pp. 857–882.

[244] S. Olver, R. M. Slevinsky, and A. Townsend. *Fast algorithms using orthogonal polynomials.* Acta Numer. 29 (2020), pp. 573–699.

[245] S. Olver, A. Townsend, and G. Vasil. *A sparse spectral method on triangles.* SIAM J. Sci. Comput. 41.6 (2019), A3728–A3756.

[246] S. Olver and A. Townsend. *A fast and well-conditioned spectral method.* SIAM Rev. 55.3 (2013), pp. 462–489.

[247] R. Orús. *A practical introduction to tensor networks: Matrix product states and projected entangled pair states.* Ann. Physics 349 (2014), pp. 117–158.

[248] I. V. Oseledets. *Approximation of matrices with logarithmic number of parameters.* Doklady Math. 428.1 (2009), pp. 23–24.

[249] I. V. Oseledets. *Tensor-train decomposition.* SIAM J. Sci. Comput. 33.5 (2011), pp. 2295–2317.

[250] I. V. Oseledets, M. V. Rakhuba, and A. Uschmajew. *Alternating least squares as moving subspace correction.* SIAM J. Numer. Anal. 56.6 (2018), pp. 3459–3479.

[251] I. Oseledets and E. Tyrtyshnikov. *TT-cross approximation for multidimensional arrays.* Linear Algebra Appl. 432.1 (2010), pp. 70–88.

[252] P. Paatero. *Construction and analysis of degenerate PARAFAC models.* J. Chemom. 14.3 (2000), pp. 285–299.

[253] B. Pass. *Multi-marginal optimal transport: Theory and applications.* ESAIM: M2AN 49.6 (2015), pp. 1771–1790.

[254] R. Peng, J. Gray, and G. Kin-Lic Chan. *Arithmetic circuit tensor networks, multivariable function representation, and high-dimensional integration.* arXiv e-prints (2022), arXiv:2209.07410.

[255] R. Penrose. *Applications of negative dimensional tensors.* In: Combin. Math. Appl. Academic Press, London, 1971, pp. 221–244.

[256] D. Perez-Garcia et al. *Matrix product state representations.* Quantum Info. Comput. 7.5 (2007), pp. 401–430.

[257] T. von Petersdorff and C. Schwab. *Numerical solution of parabolic equations in high dimensions.* M2AN Math. Model. Numer. Anal. 38.1 (2004), pp. 93–127.

[258] G. Peyré and M. Cuturi. *Computational optimal transport: With applications to data science.* Found. Trends Mach. Learn. 11.5-6 (2019), pp. 355–607.

[259] H. P. Pfeiffer et al. *A multidomain spectral method for solving elliptic equations.* Comput. Phys. Comm. 152.3 (2003), pp. 253–273.

[260] R. B. Platte and L. N. Trefethen. *Chebfun: A new kind of numerical computing.* In: Progress in Industrial Mathematics at ECMI 2008. Vol. 15. Math. Ind. Springer, 2010, pp. 69–87.

[261] R.-E. Plessix. *A Helmholtz iterative solver for 3D seismic-imaging problems.* Geophysics 72.5 (2007), SM185–SM194.

[262] T. Poggio et al. *Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review.* Int. J. Autom. Comput. 14.5 (2017).

[263] A. Popov and N. Y. Zhu. *Modeling radio wave propagation in tunnels with a vectorial parabolic equation.* IEEE Trans. Antennas Propag. 48.9 (2000), pp. 1403–1412.

[264] M. Psenka and N. Boumal. *Second-order optimization for tensors with fixed tensor-train rank.* arXiv e-prints (2020), arXiv:2011.13395.

[265] A. Qing. *Dynamic differential evolution strategy and applications in electromagnetic inverse scattering problems.* IEEE Trans. Geosci. Remote Sens. 44.1 (2006), pp. 116–125.

[266] J. Quastel. *Diffusion of color in the simple exclusion process.* Comm. Pure Appl. Math. 45.6 (1992), pp. 623–679.

[267] J. Rabin, S. Ferradans, and N. Papadakis. *Adaptive color transfer with relaxed optimal transport.* In: IEEE Int. Conf. Imag. Process. 2014, pp. 4852–4856.

[268] S. Rahnamayan, H. Tizhoosh, and M. Salama. *Opposition-based differential evolution (ODE) with variable jumping rate.* In: IEEE Symp. Found. Comput. Intell. 2007, pp. 81–88.

[269] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama. *A novel population initialization method for accelerating evolutionary algorithms.* Comput. Math. Appl. 53.10 (2007), pp. 1605–1614.

[270] P. Rai et al. *Randomized functional sparse Tucker tensor for compression and fast visualization of scientific data.* arXiv e-prints (2019), arXiv:1907.05884.

[271] M. V. Rakhuba and I. V. Oseledets. *Fast multidimensional convolution in low-rank tensor formats via cross approximation.* SIAM J. Sci. Comput. 37.2 (2015), A565–A582.

[272] S.-J. Ran et al. *Tensor Network Contractions: Methods and Applications to Quantum Many-Body Systems.* Vol. 964. Lecture Notes in Phys. Springer, 2020.

[273] E. Robeva and A. Seigal. *Duality of graphical models and tensor networks.* Inf. Inference 8.2 (2019), pp. 273–288.

[274] T. Rohwedder and A. Uschmajew. *On local convergence of alternating schemes for optimization of convex problems in the tensor train format.* SIAM J. Numer. Anal. 51.2 (2013), pp. 1134–1162.

[275] S. F. Roohi et al. *Multi-dimensional low rank plus sparse decomposition for reconstruction of under-sampled dynamic MRI.* Pattern Recognit. 63 (2017), pp. 667–679.

[276] E. Saada. *A limit theorem for the position of a tagged particle in a simple exclusion process.* Ann. Probab. 15.1 (1987), pp. 375–381.

[277] A. K. Saibaba. *HOID: Higher order interpolatory decomposition for tensors based on Tucker representation.* SIAM J. Matrix Anal. Appl. 37.3 (2016), pp. 1223–1249.

[278] A. K. Saibaba, R. Minster, and M. E. Kilmer. *Efficient randomized tensor-based algorithms for function approximation and low-rank kernel interactions.* Adv. Comput. Math. 48 (2022).

[279] E. Sanz and D. Marenduzzo. *Dynamic Monte Carlo versus Brownian dynamics: A comparison for self-diffusion and crystallization in colloidal fluids.* J. Chem. Phys. 132.19 (2010), p. 194102.

[280] S. A. Sauter and C. Schwab. *Boundary Element Methods.* Vol. 39. Springer Ser. Comput. Math. Springer, 2011.

[281] D. Savostyanov and I. Oseledets. *Fast adaptive interpolation of multi-dimensional arrays in tensor train format.* In: 7th Int. Workshop Multidimens. (nD) Syst. 2011, pp. 1–8.

[282] D. V. Savostyanov. *Quasioptimality of maximum-volume cross interpolation of tensors.* Linear Algebra Appl. 458 (2014), pp. 217–244.

[283] M. Scetbon and M. Cuturi. *Low-rank optimal transport: Approximation, statistics and debiasing.* arXiv e-prints (2022), arXiv:2205.12365.

[284] M. Scetbon, M. Cuturi, and G. Peyré. *Low-rank Sinkhorn factorization.* In: Proc. 38th Int. Conf. Mach. Learn. 2021, pp. 9344–9354.

[285] E. Schmidt. *Zur Theorie der linearen und nicht linearen Integralgleichungen Zweite Abhandlung.* Math. Ann. 64.2 (1907), pp. 161–174.

[286] R. Schneider and A. Uschmajew. *Approximation rates for the hierarchical tensor format in periodic Sobolev spaces.* J. Complexity 30.2 (2014), pp. 56–71.

[287] N. Schwenck et al. *Dimensionally reduced flow models in fractured porous media: Crossings and boundaries.* Comput. Geosci. 19.6 (2015), pp. 1219–1230.

[288] A. Shapira. *Hydrodynamic limit of the Kob-Andersen model.* arXiv e-prints (2020), arXiv:2003.08495.

[289] T. Shi and A. Townsend. *On the compressibility of tensors.* SIAM J. Matrix Anal. Appl. 42.1 (2021), pp. 275–298.

[290] N. Shigesada, K. Kawasaki, and E. Teramoto. *Spatial segregation of interacting species.* J. Theoret. Biol. 79.1 (1979), pp. 83–99.

[291] N. D. Sidiropoulos et al. *Tensor decomposition for signal processing and machine learning.* IEEE Trans. Signal Process. 65.13 (2017), pp. 3551–3582.

[292] V. de Silva and L.-H. Lim. *Tensor rank and the ill-posedness of the best low-rank approximation problem.* SIAM J. Matrix Anal. Appl. 30.3 (2008), pp. 1084–1127.

[293] N. Singh et al. *Comparison of accuracy and scalability of Gauss-Newton and alternating least squares for CANDECOMC/PARAFAC decomposition.* SIAM J. Sci. Comput. 43.4 (2021), pp. C290–C311.

[294] R. Sinkhorn. *A relationship between arbitrary positive matrices and doubly stochastic matrices.* Ann. Math. Statist. 35 (1964), pp. 876–879.

[295] S. A. Smolyak. *Quadrature and interpolation formulas for tensor products of certain classes of functions.* Dokl. Akad. Nauk SSSR 148.5 (1963), pp. 1042–1045.

[296] I. M. Sobol'. *Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates.* Math. Comput. Simulation 55.1-3 (2001), pp. 271–280.

[297] I. M. Sobol'. *Sensitivity estimates for nonlinear mathematical models.* Math. Modeling Comput. Experiment 1.4 (1993), 407–414 (1995).

[298] M. B. Soley et al. *Functional tensor-train Chebyshev method for multidimensional quantum dynamics simulations.* J. Chem. Theory Comput. 18 (2022), pp. 25–36.

[299] Z. Song, D. P. Woodruff, and P. Zhong. *Low rank approximation with entrywise ℓ_1-norm error.* In: 49th Annu. ACM Symp. Theory Comput. ACM, New York, 2017, pp. 688–701.

[300] D. C. Sorensen and M. Embree. *A DEIM induced CUR factorization.* SIAM J. Sci. Comput. 38.3 (2016), A1454–A1482.

[301] H. Spohn. *Large Scale Dynamics of Interacting Particles.* Texts Monogr. Phys. Springer, 1991.

[302] M. Stoll. *A literature survey of matrix methods for data science.* GAMM-Mitt. 43.3 (2020), e202000013, 26.

[303] J. C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations.* 2nd. SIAM, 2004.

[304] C. Strössner and D. Kressner. *Fast global spectral methods for three-dimensional partial differential equations.* IMA J. Numer. Anal. (2022), pp. 1–24.

[305] C. Strössner and D. Kressner. *Low-rank tensor approximations for solving multimarginal optimal transport problems.* SIAM J. Imaging Sci. 16.1 (2023), pp. 169–191.

[306] C. Strössner, B. Sun, and D. Kressner. *Approximation in the extended functional tensor train format.* arXiv e-prints (2022), arXiv:2211.11338.

[307] A. M. Stuart. *Inverse problems: A Bayesian perspective.* Acta Numer. 19 (2010), pp. 451–559.

[308] B. Sudret, S. Marelli, and J. Wiart. *Surrogate models for uncertainty quantification: An overview.* In: 17th Eur. Conf. Antennas Propag. 2017, pp. 793–797.

[309] A. Sultonov, S. Matveev, and S. Budzinskiy. *Low-rank nonnegative tensor approximation via alternating projections and sketching.* arXiv e-prints (2022), arXiv:2209.02060.

[310] S. Surjanovic and D. Bingham. *Virtual library of simulation experiments: Test functions and datasets.* Retrieved November 14, 2022, from https://www.sfu.ca/~ssurjano/. 2013.

[311] K. R. Swanson et al. *Virtual and real brain tumors: Using mathematical modeling to quantify glioma growth and invasion.* J. Neurol. Scie. 216.1 (2003), pp. 1–10.

[312] S. Szalay et al. *Tensor product methods and entanglement optimization for ab initio quantum chemistry.* Int. J. Quantum Chem. 115.19 (2015), pp. 1342–1391.

[313] P. R. Taylor. *Stochastic lattice-based models of diffusion in biological systems.* PhD thesis. University of Oxford, 2016.

[314] A. Thibault et al. *Overrelaxed Sinkhorn-Knopp algorithm for regularized optimal transport.* Algorithms 14.5 (2021).

[315] A. L. Thorneywork et al. *Effect of hydrodynamic interactions on self-diffusion of quasi-two-dimensional colloidal hard spheres.* Phys. Rev. Lett. 115 (26 2015), p. 268301.

[316] R. A. Todor and C. Schwab. *Convergence rates for sparse chaos approximations of elliptic problems with stochastic coefficients.* IMA J. Numer. Anal. 27.2 (2007), pp. 232–261.

[317] C. Toninelli, G. Biroli, and D. S. Fisher. *Spatial structures and dynamics of kinetically constrained models of glasses.* Phys. Rev. Lett. 92 (18 2004), p. 185504.

[318] A. Townsend. *Computing with functions in two dimensions.* PhD thesis. University of Oxford, 2014.

[319] A. Townsend and S. Olver. *The automatic solution of partial differential equations using a global spectral method.* J. Comput. Phys. 299 (2015), pp. 106–123.

[320] A. Townsend and L. N. Trefethen. *An extension of Chebfun to two dimensions.* SIAM J. Sci. Comput. 35.6 (2013), pp. C495–C518.

[321] A. Townsend, H. Wilber, and G. B. Wright. *Computing with functions in spherical and polar geometries I. The sphere.* SIAM J. Sci. Comput. 38.4 (2016), pp. C403–C425.

[322] L. N. Trefethen. *Approximation Theory and Approximation Practice.* SIAM, 2013.

[323] L. N. Trefethen. *Computing numerically with functions instead of numbers.* Math. Comput. Sci. 1.1 (2007), pp. 9–19.

[324] L. N. Trefethen. *Cubature, approximation, and isotropy in the hypercube.* SIAM Rev. 59.3 (2017), pp. 469–491.

[325] L. N. Trefethen. *Is Gauss quadrature better than Clenshaw-Curtis?* SIAM Rev. 50.1 (2008), pp. 67–87.

[326] L. N. Trefethen. *Spectral Methods in MATLAB.* Vol. 10. Software Environ. Tools. SIAM, 2000.

[327] L. R. Tucker. *Some mathematical notes on three-mode factor analysis.* Psychometrika 31 (1966), pp. 279–311.

[328] E. Ullmann, H. C. Elman, and O. G. Ernst. *Efficient iterative solvers for stochastic Galerkin discretizations of log-transformed random diffusion problems.* SIAM J. Sci. Comput. 34.2 (2012), A659–A682.

[329] C. Vanaret et al. *Certified global minima for a benchmark of difficult optimization problems.* arXiv e-prints (2020), arXiv:2003.09867.

[330] M. Vandecappelle, N. Vervliet, and L. De Lathauwer. *A second-order method for fitting the canonical polyadic decomposition with non-least-squares cost.* IEEE Trans. Signal Process. 68 (2020), pp. 4454–4465.

[331] F. Verstraete, V. Murg, and J. Cirac. *Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems.* Adv. Phys. 57.2 (2008), pp. 143–224.

[332] N. Vervliet et al. *Tensorlab 3.0.* 2016.

[333] C. Villani. *Optimal Transport: Old and New.* 1st. Grundlehren Math. Wiss. Springer, 2009.

[334] J. Waldvogel. *Fast construction of the Fejér and Clenshaw-Curtis quadrature rules.* BIT 46.1 (2006), pp. 195–202.

[335] W. Wang et al. *A linear optimal transportation framework for quantifying and visualizing variations in sets of images.* Int. J. Comput. Vis. 101.2 (2013), pp. 254–269.

[336] Z. Wang and S. F. Wu. *Helmholtz equation–least-squares method for reconstructing the acoustic pressure field.* J. Acoust. Soc. Am. 102.4 (1997), pp. 2020–2032.

[337] S. R. White. *Density matrix formulation for quantum renormalization groups.* Phys. Rev. Lett. 69 (19 1992), pp. 2863–2866.

[338] H. Wilber, A. Townsend, and G. B. Wright. *Computing with functions in spherical and polar geometries II. The disk.* SIAM J. Sci. Comput. 39.3 (2017), pp. C238–C262.

[339] D. P. Woodruff. *Sketching as a tool for numerical linear algebra.* Found. Trends Theor. Comput. Sci. 10.1-2 (2014), pp. iv+157.

[340] H. Xiang and L. Grigori. *Kronecker product approximation preconditioners for convection-diffusion model problems.* Numer. Linear Algebra Appl. 17.4 (2010), pp. 691–712.

[341] D. Xiu. *Stochastic collocation methods: A survey.* In: Handbook of Uncertainty Quantification. Springer, 2017, pp. 699–716.

[342] H. Xu et al. *Spectral/hp element methods: Recent developments, applications, and perspectives.* J. Hydrodyn. 30.1 (2018), pp. 1–22.

[343] N. L. Zamarashkin, S. V. Morozov, and E. E. Tyrtyshnikov. *On the best approximation algorithm by low-rank matrices in Chebyshev's norm.* Comput. Math. Math. Phys. 62.5 (2022), pp. 701–718.

[344] V. P. Zankin, G. V. Ryzhakov, and I. V. Oseledets. *Gradient descent-based D-optimal design for the least-squares polynomial approximation.* arXiv e-prints (2018), arXiv:1806.06631.

[345] C. Zenger. *Sparse grids.* In: Parallel Algorithms for Partial Differential Equations. Vol. 31. Notes Numer. Fluid Mech. Friedr. Vieweg & Sohn, 1991, pp. 241–251.

[346] S. Zhao and M. J. Yedlin. *A new iterative Chebyshev spectral method for solving the elliptic equation* $\nabla \cdot (\sigma \nabla u) = f$. J. Comput. Phys. 113.2 (1994), pp. 215–223.

[347] M. S. Zhdanov, S. K. Lee, and K. Yoshioka. *Integral equation method for 3D modeling of electromagnetic fields in complex structures with inhomogeneous background conductivity.* Geophysics 71.6 (2006), G333–G345.

[348] B. Zhou and M. Parno. *Efficient and exact multimarginal optimal transport with pairwise costs.* arXiv e-prints (2022), arXiv:2208.03025.

[349] G. Zhou et al. *Efficient nonnegative Tucker decompositions: Algorithms and uniqueness.* IEEE Trans. Image Process. 24.12 (2015), pp. 4990–5003.

[350] Y. Zhou et al. *Large-scale parallel collaborative filtering for the Netflix prize.* In: Algorithmic Aspects in Information and Management. Springer, 2008, pp. 337–348.

Curriculum Vitae

Personal Details

Christoph Max Strößner
born 22.09.1995 in Hof/Germany

Education

École Polytechique Fédérale de Lausanne Switzerland
PhD in Mathematics since 01/2019
Thesis: Low-Rank Tensor Methods for High-Dimensional Problems
Advisor: Prof. Daniel Kressner

Technical University of Munich Germany
Master of Science in Mathematics 10/2016-09/2018
Thesis: A Surrogate Model for Bayesian Inversion Based on Sparse Grids
Thesis advisor: Prof. Barbara Wohlmuth

Technical University of Munich Germany
Bachelor of Science in Mathematics 10/2013-09/2016
Thesis: An Optimization Algorithm for Finding Invariant Sets
Thesis advisor: Prof. Oliver Junge

KAIST South-Korea
Semester abroad 08/2015-12/2015

Professional Experience

École Polytechique Fédérale de Lausanne Switzerland
Research Assistant since 01/2019

Technical University of Munich Germany
Student Assistant 09/2016-01/2018

ESG Elektroniksystem- und Logistik-GmbH Germany
Research and Development Intern 01/2016-03/2016

Publications and Preprints

C. Strössner and D. Kressner, *Low-rank tensor approximations for solving multimarginal optimal transport problems.* SIAM J. Imaging Sci. 2023 16:1, 169–191. DOI. GitHub.

C. Strössner and D. Kressner, *Fast global spectral methods for three-dimensional partial differential equations.* IMA J. Numer. Anal. 2022. DOI. GitHub.

S. Dolgov, D. Kressner and C. Strössner, *Functional Tucker approximation using Chebyshev interpolation*, SIAM J. Sci. Comput. 2021 43:3, A2190-A2210. DOI. GitHub.

J. Dabaghi, V. Ehrlacher and C. Strössner, *Computation of the self-diffusion coefficient with low-rank tensor methods: application to the simulation of a cross-diffusion system*, To appear in ESAIM Proc. Surveys. arXiv. GitHub.

J. Dabaghi, V. Ehrlacher and C. Strössner, *Tensor approximation of the self-diffusion matrix of tagged particle processes.* To appear in J. Comput. Phys.. arXiv. GitHub.

C. Strössner, B. Sun and D.Kressner, *Approximation in the extended functional tensor train format.* Submitted. arXiv. GitHub.

Awards

Dean's Award for Excellence in Teaching 2021-22

Scholarships

Max Weber Program

KAIST Scholarship Award

Memberships

SIAM member since 2021

Conferences, Seminars, Summer Schools and Workshops

Workshop on Low-rank Models and Applications 22 Mons
Talk: Low-rank tensor approximations for solving multi-marginal optimal transport problems 2022

Swiss Numerics Day 2022 Zurich
Poster: Low-rank multi-marginal optimal transport 2022

ApplMath22 Brijuni
Poster: Low-rank multi-marginal optimal transport 2022

MATHICSE Retreat Villars-sur-Ollon
Talk: Functional low-rank approximations based on tensorized Chebyshev interpolation 2022

CERMICS Applied Mathematics Seminar Marne-la-Vallée
Invited Talk: Functional low-rank approximations for trivariate functions based on tensorized Chebyshev polynomials 2021

Joint Group Day 2021 Lausanne
Talk: Functional Tucker approximations: Novel constructors and solvers 2021

Swiss Numerics Day 2021 Lausanne
2021

Matrix Equations and Tensor Techniques IX Perugia
Talk: Solving PDEs on hypercubes using Chebyshev interpolation 2021

CEMRACS Research Project Luminy
Talk: Computation of the self-diffusion coefficient of a cross-diffusion system with tensor methods. 2021

CEMRACS 2021 Summer School Luminy
2021

SIAM Conference on Applied Linear Algebra (LA21) virtual
2021

GAMM 2020@21 virtual
Invited Talk: Functional Tucker approximation using Chebyshev interpolation 2021

Communications in NLA virtual
Talk: Functional Tucker approximation using Chebyshev interpolation 2021

Low-Rank Models 2020 Villars-sur-Ollon
Poster: Chebychev interpolation in Tucker format 2020

Model Order Reduction Summer School Eindhoven
2019

8th Workshop on High-Dimensional Approximation Zürich
2019

MATHICSE Retreat Champéry
Talk: Interpolation based on low-rank tensors 2019

Swiss Numerics Day 2019	Lugano 2019
Winter School on Hierarchical Matrices	Kiel 2019

Teaching Assistantship at EPFL

Analyse IV	Spring 2022
Analysis 1	Fall 2021
Numerical Analysis	Spring 2021
Analysis 1	Fall 2020
Probabilities and Statistics II	Spring 2020
Discrete Mathematics	Fall 2019
Probabilités et Statistique	Spring 2019

Co-supervised Semester Projects at EPFL

Algorithms for image deconvolution	Spring 2022
Functional tensor train approximation using Chebyshev interpolation	Fall 2021
Nonnegative low-rank matrix approximation for nonnegative matrices	Spring 2021
Mixed precision algorithms for fast matrix factorizations	Fall 2020
The Sinkhorn algorithm for optimal transport	Fall 2020