The Tensor-Train Format and Its Applications

Modeling and Analysis of Chemical Reaction Networks,
Catalytic Processes, Fluid Flows, and Brownian Dynamics

Dissertation zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

eingereicht im Fachbereich Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von

Patrick Gelß

Berlin 2017

Erstgutachter: Prof. Dr. Christof Schütte
Freie Universität Berlin
Fachbereich Mathematik und Informatik
Arnimallee 6
14195 Berlin

Zweitgutachter: Prof. Dr. Reinhold Schneider
Technische Universität Berlin
Institut für Mathematik
Straße des 17. Juni 136
10623 Berlin

Tag der Disputation: 28. Juni 2017

Abstract

The simulation and analysis of high-dimensional problems is often infeasible due to the curse of dimensionality. In this thesis, we investigate the potential of tensor decompositions for mitigating this curse when considering systems from several application areas. Using tensor-based solvers, we directly compute numerical solutions of master equations associated with Markov processes on extremely large state spaces. Furthermore, we exploit the tensor-train format to approximate eigenvalues and corresponding eigentensors of linear tensor operators. In order to analyze the dominant dynamics of high-dimensional stochastic processes, we propose several decomposition techniques for highly diverse problems. These include tensor representations for operators based on nearest-neighbor interactions, construction of pseudoinverses for tensor-based reformulations of dimensionality reduction methods, and the approximation of transfer operators of dynamical systems. The results show that the tensor-train format enables us to compute low-rank approximations for various numerical problems as well as to reduce the memory consumption and the computational costs compared to classical approaches significantly. We demonstrate that tensor decompositions are a powerful tool for solving high-dimensional problems from various application areas.

Acknowledgements

I would like to take this opportunity to express my gratitude to all those who encouraged me to write this thesis. First and foremost, I would like to thank my supervisor Christof Schütte for his continuous support and guidance as well as for offering me the possibility of writing this thesis. I wish to express my sincere appreciation to Stefan Klus for proofreading this thesis and for providing valuable comments and suggestions. I have greatly benefited from his support and advice. My gratitude also goes to Sebastian Matera for drawing my attention to various application areas for tensors since the beginning of my PhD.

Special thanks should be given to Thomas von Larcher for providing me with CFD data and to Sebastian Peitz for his visualizations. Additionally, I want to thank all the people from the Biocomputing Group at the FU Berlin, the members of the CRC 1114, and the research group around Reinhold Schneider at TU Berlin for the interesting discussions and valuable inputs.

Finally, I would like to thank my family and friends for their support during the preparation of this work. In particular, I am deeply grateful to Nadja. Without her love and help during the last years this all would not have been possible.

This research has been funded by the Berlin Mathematical School and the Einstein Center for Mathematics.

To my son, Finn.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ALS	alternating linear scheme (cf. §4.1ff.)
BTT	block tensor train (cf. §3.5.2)
CFD	computational fluid dynamics (cf. §11.1)
CME	chemical master equation (cf. §5.2, §9.2)
CO	carbon monoxide (cf. §10.2)
CO_2	carbon dioxide (cf. §10.2)
CRN	chemical reaction network (cf. §9.1)
CTT	cyclic tensor train (cf. §3.5.3)
cus	coordinatively unsaturated site (cf. §10.2)
DMD	dynamic mode decomposition (cf. §7.3)
HT	hierarchical Tucker (cf. §3.3)
kMC	kinetic Monte Carlo (cf. §10.1)
MALS	modified alternating linear scheme (cf. §4.1ff.)
MME	Markovian master equation (cf. §5.1)
NNIS	nearest-neighbor interaction system (cf. §6.1)
O	oxygen (cf. §10.2)
ODE	ordinary differential equation (cf. §4.5)
QTT	quantized tensor train (cf. §3.5.1)
RuO_2	ruthenium dioxide (cf. §10.2)
SCR	single-cell reaction (cf. §6.3)
SDE	stochastic differential equation (cf. §12.1)
SVD	singular value decomposition (cf. §2.6)
TCR	two-cell reaction (cf. §6.3)
TDMD	tensor-based dynamic mode decomposition (cf. §7.3)
TOF	turn-over frequency (cf. §10.3.2)
TT	tensor train (cf. §3.4ff.)

List of Symbols

Symbols

$\langle\,.\,,\,.\,\rangle$	Euclidean inner product
$\langle\,.\,,\,.\,\rangle_{\bullet,\ldots,\bullet}$	index contraction (cf. §2.2.2)
$[\,.\,]$	core notation (cf. §3.4.1)
$\overline{\bullet}$	operator expression of a tensor (cf. §2.1)
$\overline{\bullet,\ldots,\bullet}$	multi-index notation (cf. §2.4)
$\bullet^T$	transpose of a matrix or a tensor (cf. §2.1)
$\bullet^{\mathbb{T}}$	rank-transpose of TT core (cf. §3.4.1)
$\bullet^{-1}$	inverse of a matrix or inverse function
$\bullet\Big\vert_{\bullet}^{\bullet}$	matricization of a tensor (cf. §2.4)
$\times$	Cartesian product of sets
$\otimes, \bigotimes$	tensor product of two or more tensors (cf. §2.2.4)
$\|\,.\,\|_p$	p-norm of a vector or tensor (cf. §2.5)
$\|\,.\,\|_F$	Frobenius norm of a matrix or tensor operator (cf. §2.5)

Greek Letters

$\delta_{i,j}$	Kronecker delta with respect to natural numbers i and j
Θ_i	cell of an NNIS (cf. §6.1)
Σ	diagonal matrix of an SVD
τ	step size
ϕ_N	little-endian bijection (cf. §2.4)
$\xi_\mu, \xi_{i,\mu}, \xi_{i,i+1,\mu}$	vectors of net changes (cf. §5.2, §6.3)

Latin Letters

$\mathbf{A}$, $\mathbf{G}$, $\mathbf{H}$	linear tensor operators (cf. §2.1)
$\mathbf{A}^{(i)}$, $\mathbf{G}^{(i)}$, $\mathbf{H}^{(i)}$	TT cores of linear tensor operators (cf. §3.4)
A, U, V	matrices
a_μ	reaction propensity (cf. §5.2)
$\mathbf{a}_\mu$, $\mathbf{a}_{i,\mu}$, $\mathbf{a}_{i,i+1,\mu}$	propensity tensors (cf. §5.2, §6.3)
d	order of a tensor (cf. §2.1)
$e_{\mathrm{IE},k}$	residual error for implicit Euler method (cf. §4.5)
$e_{\mathrm{TR},k}$	residual error for trapezoidal rule (cf. §4.5)

e_λ, e_{λ_k}	approximation errors for eigenvalues (cf. §11.2.1)
e_φ, e_{φ_k}	approximation errors for eigentensors (cf. §11.2.1)
$\mathbf{G}_\mu$, $\mathbf{G}_{i,\mu}$, $\mathbf{G}_{i,i+1,\mu}$	multidimensional shift operator (cf. §5.2, §6.3)
$G_i(\,.\,)$	shift matrix (cf.§5.2)
I	identity matrix
$\Im$	imaginary part of a complex number
$\mathbf{I}$	identity tensor (cf. §2.1)
$\mathcal{L}(\,.\,)$	left-unfolding of a TT core (cf. §3.4.3)
M, N, P	index sets (cf. §2.1)
$\mathrm{mat}(\,.\,)$	natural matricization (cf. §2.4)
$\mathbb{N}$	set of natural numbers $\{1, 2, \dots\}$
$\mathbb{N}_0$	set $\mathbb{N} \cup \{0\} = \{1, 2, \dots\}$
$\mathbb{N}^d$, $\mathbb{N}_0^d$	vectors with elements in $\mathbb{N}$ and $\mathbb{N}_0$, respectively
$O(\,.\,)$	Landau symbol
$P(X, t)$	probability of being in state X at time t (cf. §5.1)
$\mathbf{P}(t)$	probability tensor (cf. §5.2)
r, R	ranks of tensor decompositions
R_μ, $\mathrm{R}_{i,\mu}$, $\mathrm{R}_{i,i+1,\mu}$	elementary reactions (cf. §5.2, §6.3)
$\mathcal{R}(\,.\,)$	right-unfolding of a TT core (cf. §3.4.3)
$\mathbb{R}$	field of real numbers
$\mathbb{R}^m$, $\mathbb{R}^n$	vectors with elements in $\mathbb{R}$
$\mathbb{R}^{m\times n}$	real matrices with m rows and n columns
$\mathbb{R}^M$, $\mathbb{R}^N$	tensor spaces (cf. §2.1)
$\mathbb{R}^{M\times N}$, $\mathbb{R}^{N\times P}$	spaces of linear tensor operators (cf. §2.1)
$\mathcal{S}$	state space
$\mathbf{T}, \mathbf{U}, \mathbf{V}$	tensors
$\mathbf{T}^{(i)}$, $\mathbf{U}^{(i)}$, $\mathbf{V}^{(i)}$	TT cores of tensor trains (cf. §3.4)
$\mathrm{tr}(\,.\,)$	trace of a tensor in CTT format (cf. §3.5.3)
v, w	vectors
$\mathrm{vec}(\,.\,)$	natural vectorization (cf. §2.4)
X	state of a system

1 Introduction

Over the last years, low-rank tensor approaches have become an important tool for the mathematical modeling and numerical simulation of high-dimensional systems as well as for the approximation of high-dimensional functions. Tensor-based methods have been successfully used in many different application areas such as quantum physics [1, 2], chemical reaction dynamics [3, 4, 5, 6], stochastic queuing problems [7, 8, 9], machine learning [10, 11, 12], and high-dimensional data analysis [13, 14]. In our sense, tensors are viewed as multidimensional generalizations of matrices, represented by arrays with several indices. The number of elements of these tensors grows exponentially with the number of dimensions, and so does the storage consumption. This phenomenon is referred to as the *curse of dimensionality*. The interest in tensor decompositions has been growing rapidly within the scientific computing community as recently developed formats for the representation of tensors in form of tensor networks [15, 16] have shown that it is possible to mitigate the curse of dimensionality and to tackle high-dimensional systems and large-scale problems which could not be analyzed by conventional numerical methods before. That is, different tensor formats such as the *tensor-train format* (TT format) [17] enable the simulation and analysis of high-dimensional problems without an exponential scaling of the memory consumption and the computational complexity with the number of dimensions. Typically, the applications require the approximation of the solutions of systems of linear equations, eigenvalue problems, ordinary/partial differential equations, or completion problems, see e.g. [9, 18, 19, 20]. The aim is to carry out all numerical computations directly in suitable tensor formats without the decomposition of tensors in full format.

There are different opinions on the origin of the tensor concept, cf. [21]. In 1846, the word "tensor" (latin: tendere, tensus – to stretch/spread [22]) was first used in a mathematical sense by William Rowan Hamilton [23], introducing the word as a term for the norm of a quaternion. Half a decade later, Woldemar Voigt was the first who related tensors to the contemporary meaning as a generalization of scalars, vectors and matrices [24]. Interestingly, Josiah Willard Gibbs extended Hamilton's idea already in the early 1880s and considered linear vector functions which he called dyadics. Not until 1901 did he publish his ideas together with Edwin Bidwell Wilson [25], introducing a concept that came very close to the theory of tensors as we understand it nowadays. However, the origin of tensors may also be traced back to the field of differential geometry during the 19th century, including the work of Carl Friedrich Gauß [26], Elwin Bruno Christoffel [27], Gregorio Ricci–Curbastro [28], and Tullio Levi–Civita [29]. Their concepts then also played an important role in the work of James Clerk Maxwell [30, 31], resulting in what we call Maxwell's

stress tensors. Eventually, the framework of tensors received broader acceptance around 1916 when Albert Einstein published the general theory of relativity [32] using the language of tensors.

The foundation of tensor decompositions is provided by the so-called *tensor product*, which enables us to decompose high-dimensional tensors into several smaller tensors. The simplest form of tensor decompositions is given by so-called *rank-one tensors*, i.e. the representation of a tensor as the tensor product of a set of vectors. Extending the concept of rank-one tensors, the main idea of more complex tensor decompositions is the representation of high-dimensional tensors as a network of low-dimensional tensors coupled by so-called *ranks*. These ranks have a strong influence on the capability of representing a given tensor as well as on the required memory. The initial concept of tensor decompositions was introduced in 1927 by Frank Hitchcock who presented the idea of expressing a tensor as the sum of a finite number of rank-one tensors [33] – the so-called *canonical format*, also abbreviated as CANDECOMP [34] and PARAFAC for *Parallel Factor Analysis* [35]. Unfortunately, even though the canonical format would be optimal in terms of memory consumption, it is numerically unstable [36, 37]. In 1963, Ledyard Tucker introduced the *Tucker format* [38, 39], which is known in quantum chemistry in the context of the *multiconfiguration time-dependent Hartree method* [40]. On the one hand, tensors represented in the Tucker format with fixed ranks form an embedded manifold [41] and, thus, we can rely on robust algorithms. On the other hand, the storage consumption of Tucker tensors depends exponentially on the number of dimensions. One of the most promising tensor formats is the so-called TT format developed by Ivan Oseledets and Eugene Tyrtyshnikov in 2009, see [16, 17, 42]. It is a special case of the almost simultaneously proposed *Hierarchical Tucker Format* (HT format) [43, 44], which combines the advantages of the canonical format and the Tucker format, i.e. the storage consumption of a tensor in HT format does not depend exponentially on the number of dimensions and there exist robust algorithms for the computation of best approximations. In quantum physics, the TT format is known as *matrix product state representation* and was already introduced in 1987 [45]. However, the concept of tensor decompositions and approximations is rather new in the field of numerical mathematics, providing the opportunity for broad theoretical and experimental research. For an overview of different low-rank tensor approximation approaches, we refer to [46]. In this work, we will particularly focus on linear tensor operators in the TT format defined on extremely large state spaces and the solution of corresponding systems of linear equations or eigenvalue problems. Algorithms for solving such systems in the TT format are, for instance, the *alternating linear scheme* (ALS) and the *modified alternating linear scheme* (MALS), see [37]. The basic idea is to fix all components of the tensor network except for one. This yields a series of low-dimensional problems, which can then be solved using classical numerical methods. The efficiency of the tensor-based algorithms depends strongly on the TT ranks of the operator. Thus, it is important to be able to find low-rank representations of a given tensor, which is one of the most challenging tasks in tensor-based problem formulations.

In order to understand the dynamical properties of stochastic processes on high-dimensional state spaces, one can describe many systems by *Markov processes* which

are continuous in time and discrete in space, see e.g. [47]. A jump then corresponds to the execution of a particular event changing the state of the system. In practice, the state spaces of these models may be extremely high-dimensional making it impossible to solve the corresponding *Markovian master equation* (MME) [48] by using classical numerical methods. A common approach to circumvent the curse of dimensionality is the application of *Monte Carlo methods* [49, 50, 51], which simulate trajectories of the stochastic processes and estimate considered quantities by statistical averaging. However, a drawback of Monte Carlo methods is the large number of simulations needed to capture relevant dynamics. Because of this limitation, we exploit the TT format in order to numerically approximate the solution of the master equation directly by using implicit integration schemes such as the implicit Euler method or the trapezoidal rule.

In particular, we consider interaction networks described by an MME that can be written in a tensor-based notation and solved by the methods explained above. In [9], we derived systematic TT decompositions for high-dimensional systems based on nearest-neighbor interactions, which can represent highly diverse physical or biological systems, e.g. coupled laser arrays [52], n-body dynamics [53], and chemical reaction networks [49]. With the aid of these decompositions, we can reduce the storage consumption as well as the computational effort significantly. We have also shown that the rank of the corresponding TT operator does not depend on the number of dimensions in some cases. Thus, the storage consumption as well as the computational complexity scale linearly with the system size.

As we presented in [6] and [9], TT decompositions for nearest-neighbor interaction systems can be used to model processes from the field of heterogeneous catalysis, which is a key technology for sustainable energy conversion and modern reaction technologies, see e.g. [54, 55]. We considered a reduced model for the CO oxidation at a catalytic surface and computed stationary probability distributions over the possible surface configurations in order to investigate the catalytic efficiency under various conditions. In our experiments, we saw that the TT approach provides high numerical accuracy over a large range of input parameters for the model and shows a better scaling behavior than Monte Carlo methods for a sequence of problems with increasing stiffness. Thus, mitigating the curse of dimensionality by using low-rank TT representations, we may be able to understand the interplay of the elementary surface reactions making up different catalytic cycles.

Another application of the TT format is the extension of the *dynamic mode decomposition* (DMD). Introduced by Peter Schmid et al. in 2008 [56, 57], DMD can be used to analyze complex dynamical systems by decomposing high-dimensional data into coupled spatial-temporal modes [58]. Assuming a linear relationship between different snapshots of the system, DMD computes eigenvalues and corresponding eigenvectors, which often represent coherent structures, for instance, in flow fields. We proposed a tensor-based dynamic mode decomposition in [14], where we showed how to construct pseudoinverses of given tensor trains without approximation in order to compute the DMD modes directly in the TT format. With the aid of several fluid dynamics problems such as the Kármán vortex street [59], we illustrated the efficiency of the TT approach.

The global behavior of dynamical systems can also be analyzed by computing

the eigenvalues and corresponding eigenfunctions of linear transfer operators [60] associated with the system. One important operator which is frequently used to gain insight into the system's behavior is the Perron–Frobenius operator [61, 62]. Eigenfunctions of this operator can be used to understand the long-term behavior of a dynamical system and to detect metastable sets in the state space. By using certain disretization techniques, the eigenfunctions can be approximated by the eigenvectors of a finite-dimensional counterpart of the Perron–Frobenius operator. However, approximating eigenfunctions of high-dimensional transfer operators is in general infeasible due to the curse of dimensionality. Using (M)ALS in combination with a modified TT format, we will show that the use of low-rank tensor approximations potentially enables the computation of these eigenfunctions.

This thesis is organized in three parts. In Part I, we will introduce the framework for tensor decompositions. Tensors in full format are described Chapter 2, several tensor formats are explained in Chapter 3, and the background on optimization problems in the TT format is provided in Chapter 4. In Part II, we will present our own contributions to the concept of TT decompositions based on our publications [6, 9, 14], see Chapters 5–7. Furthermore, we show the first steps towards the approximation of transfer operators and their eigenfunctions using (M)ALS in Chapter 8. We will also illustrate the performance of the different tensor-based approaches by considering several examples from various application areas in Part III. That is, we will consider chemical reaction networks in Chapter 9, repeat our experiments for a heterogeneous catalytic process from [6] in Chapter 10, and show examples for fluid and molecular dynamics in Chapters 11 and 12, respectively. We will conclude with a brief summary and possibilities for further research in Chapter 13.

Part I

Foundations of Tensor Approximation

"Doubtless we cannot see that other higher Spaceland
now, because we have no eye in our stomachs."

Edwin A. Abbott,
Flatland: A Romance of Many Dimensions

Part I of this thesis will focus on the theoretical foundations of tensor decompositions. In Chapter 2, we will introduce tensors in general. Our considerations will include basic definitions and mathematical operations as well as graphical representations of tensors. Important concepts such as matricizations and orthonormality of tensors will also be presented. Various tensor formats will be explained in Chapter 3. In particular, we will focus on the tensor-train format and modified versions of it. In Chapter 4, we will then consider optimization tasks in these formats and give algorithms for solving systems of linear equations and eigenvalue problems.

2
Tensors in Full Format

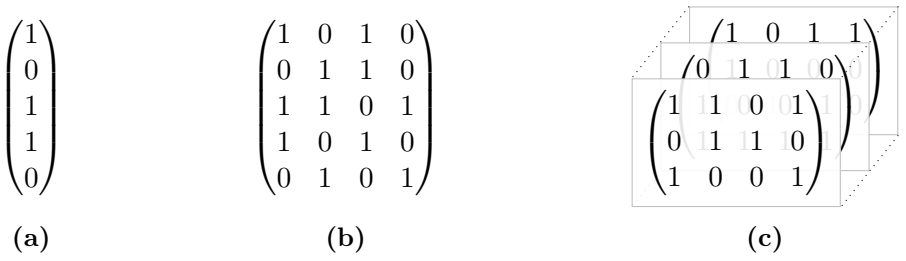
In this chapter, we will introduce the theoretical framework of tensors and clarify the notational conventions which will later be used in the context of tensor decompositions and approximations. We will give an overview of basic mathematical operations for tensors that arise from generalizing standard matrix addition and multiplication. Furthermore, we will introduce index contractions and the tensor product which will be necessary in order to develop the different tensor formats appearing in this work. We will also present a graphical representation of tensors that we will use at several points to visualize tensor operations. An important tool in the context of tensor decompositions are matricizations and vectorizations of tensors, which will be described subsequently. At the end of this chapter, we will use matricizations and vectorizations to define norms for tensors and to generalize the concept of orthonormality.

2.1. Definition and Notation

There are different approaches for defining tensors in mathematics and physics. In this work, tensors are viewed as multidimensional generalizations of matrices, represented by arrays with d indices:

$$\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}, \tag{2.1.1}$$

where $n_i \in \mathbb{N}$ for $i = 1, \ldots, d$, $d \in \mathbb{N}$. The different dimensions n_i of the array are called *modes* and the total number of modes d is called the *order* of the tensor. Examples for low-dimensional tensors are shown in Figure 2.1.



(a) **(b)** **(c)**

Figure 2.1: *Low-dimensional tensors represented by arrays: (a) A tensor of order 1 is a vector. (b) A tensor of order 2 is a matrix. (c) A tensors of order 3 can be visualized as layers of matrices.*

Throughout this work, we will denote tensors by bold letters and refer to an element of a tensor $\mathbf{T}$ by using subscript indices, i.e. $\mathbf{T}_{x_1,\dots,x_d} \in \mathbb{R}$ with $1 \leq x_i \leq n_i$ for $i = 1, \dots, d$. For a more compact notation, we define

$$\mathbb{R}^N = \mathbb{R}^{n_1 \times \cdots \times n_d}, \tag{2.1.2}$$

with the *mode set* or *index set* $N = (n_1, \dots, n_d)^T \in \mathbb{N}^d$. On the one hand, we will treat mode sets as vectors of natural numbers to point out the ordering, on the other hand, we will consider unions and intersections of mode sets where we do not pay attention to the orderings. At the respective passages in the text, this will be indicated. Note that the ordering of N matters and any reordering $N' = (n_{j_1}, \dots, n_{j_d})^T$ of N would induce a reordering of the elements of $\mathbf{T} \in \mathbb{R}^N$ into a new tensor $\mathbf{T}' \in \mathbb{R}^{N'}$.

The storage consumption of a (non-sparse) tensor of the form (2.1.1) can be estimated as $O(n^d)$, where n is the maximum of all mode sizes $n_1, \dots, n_d$. That is, the number of elements of a tensor grows exponentially with the order. Due to this so-called *curse of dimensionality*, storing a d-dimensional tensor may be infeasible for growing d. Therefore, we require special representations and approximations of tensors, which we will introduce in the next chapter.

In this work, we also consider linear operators $\mathbf{G}$ acting on tensor spaces with the same order:

$$\mathbf{G} : \mathbb{R}^N \to \mathbb{R}^M, \quad \mathbf{T} \mapsto \mathbf{G} \cdot \mathbf{T},$$

with $N = (n_1, \dots, n_d)^T$ and $M = (m_1, \dots, m_d)^T$. Tensor operators $\mathbf{G}$ are multidimensional generalizations of matrices with pairs of modes. We define these operators as tensors in

$$\mathbb{R}^{M \times N} = \mathbb{R}^{(m_1 \times n_1) \times \cdots \times (m_d \times n_d)}. \tag{2.1.3}$$

Note that other definitions of $\mathbb{R}^{M \times N}$ by reordering the modes $m_1, \dots, m_d$ and $n_1, \dots, n_d$ are also possible. It will later become clear why the order of the modes as given in (2.1.3) is justified. Similar to the standard matrix-vector product, the tensor $(\mathbf{G} \cdot \mathbf{T}) \in \mathbb{R}^M$ is given by

$$(\mathbf{G} \cdot \mathbf{T})_{x_1,\dots,x_d} := \sum_{y_1=1}^{n_1} \cdots \sum_{y_d=1}^{n_d} \mathbf{G}_{x_1,y_1,\dots,x_d,y_d} \cdot \mathbf{T}_{y_1,\dots,y_d}. \tag{2.1.4}$$

Equation (2.1.4) represents a special case of so-called index contractions, see Section 2.2.2.

Example 2.1.1. *The identity tensor* $\mathbf{I} \in \mathbb{R}^{N \times N}$ *with* $N = (n_1, \dots, n_d)^T \in \mathbb{N}^d$ *is*

defined by

$$\mathbf{I}_{x_1,y_1,\ldots,x_d,y_d} = \delta_{x_1,y_1} \cdot \ldots \cdot \delta_{x_d,y_d},$$

where δ_{x_i,y_i}, $i = 1,\ldots,d$, *denotes the Kronecker delta.*

For several of the following theorems, we exploit the fact that we can associate a tensor $\mathbf{T} \in \mathbb{R}^N$, $N = (n_1,\ldots,n_d)^T \in \mathbb{N}^d$, with a tensor $\overline{\mathbf{T}} \in \mathbb{R}^{N\times\mathbb{1}}$, $\mathbb{1} = (1,\ldots,1)^T \in \mathbb{N}^d$, by defining

$$\overline{\mathbf{T}}_{x_1,1,\ldots,x_d,1} = \mathbf{T}_{x_1,\ldots,x_d}. \tag{2.1.5}$$

The transpose of an operator $\mathbf{G} \in \mathbb{R}^{M\times N}$ is given by the tensor $\mathbf{G}^T \in \mathbb{R}^{N\times M}$ with

$$\mathbf{G}^T_{x_1,y_1,\ldots,x_d,y_d} = \mathbf{G}_{y_1,x_1,\ldots,y_d,x_d}. \tag{2.1.6}$$

The transpose of a tensor $\mathbf{T} \in \mathbb{R}^N$ is then given by $\mathbf{T}^T \in \mathbb{R}^{\mathbb{1}\times N}$ with

$$\mathbf{T}^T = \overline{\mathbf{T}}^T. \tag{2.1.7}$$

If we fix certain indices, colons are used to indicate the free modes (cf. MATLAB colon notation), e.g. for a tensor $\mathbf{T} \in \mathbb{R}^N$, $N = (n_1,\ldots,n_d)^T$, we obtain

$$\mathbf{T}_{x_1,:,x_3,:,x_5,\ldots,x_d} \in \mathbb{R}^{n_2\times n_4} \quad \text{and} \quad \mathbf{T}_{x_1,:,\ldots,:,x_d} \in \mathbb{R}^{n_2\times\cdots\times n_{d-1}}.$$

2.2. Tensor Calculus

In this section, we will give an overview of basic mathematical operations for tensors. We will first introduce the addition of tensors and the multiplication by scalar values, which both are analogous to the addition and scalar multiplication in the matrix and vector case. After that, we will consider index contractions of tensors in general and the multiplication of two tensors in particular. We will see that many operations from standard linear algebra can be easily adapted to tensors. At the end of this section, we will define the tensor product, which provides the basis for tensor decompositions and approximations introduced in the next chapter.

2.2.1. Addition and Scalar Multiplication

The sum of two tensors $\mathbf{T}$ and $\mathbf{U}$ in the same tensor space $\mathbb{R}^N$, $N = (n_1,\ldots,n_d)^T$, is a tensor whose entries are computed by adding corresponding elements.

Definition 2.2.1. *For tensors* $\mathbf{T} \in \mathbb{R}^N$ *and* $\mathbf{U} \in \mathbb{R}^N$, $N = (n_1, \ldots, n_d)^T \in \mathbb{N}^d$, *the sum* $\mathbf{T} + \mathbf{U}$ *is given by*

$$(\mathbf{T} + \mathbf{U})_{x_1,\ldots,x_d} = \mathbf{T}_{x_1,\ldots,x_d} + \mathbf{U}_{x_1,\ldots,x_d}. \tag{2.2.1}$$

The multiplication by a scalar is also performed elementwise, equivalent to the multiplication of a matrix or a vector by a scalar.

Definition 2.2.2. *For a tensor* $\mathbf{T} \in \mathbb{R}^N$, $N = (n_1, \ldots, n_d)^T \in \mathbb{N}^d$, *and a scalar* $\lambda \in \mathbb{R}$, *the product* $\lambda \cdot \mathbf{T}$ *is given by*

$$(\lambda \cdot \mathbf{T})_{x_1,\ldots,x_d} = \lambda \cdot \mathbf{T}_{x_1,\ldots,x_d}. \tag{2.2.2}$$

It is easy to prove that the addition is commutative and associative. Furthermore, the distributive property holds for the scalar multiplication. Thus, together with the above definitions, $\mathbb{R}^N$ forms a linear space in the classical sense.

2.2.2. Index Contraction

An operation that we will frequently use in this work is the contraction of one or more common indices of two given tensors. Let $\mathbf{T}$ and $\mathbf{U}$ be two tensors sharing a set of indices, i.e.

$$\mathbf{T} \in \mathbb{R}^{m_1 \times \ldots \times m_d \times p_1 \times \ldots \times p_f},$$
$$\mathbf{U} \in \mathbb{R}^{n_1 \times \ldots \times n_e \times p_1 \times \ldots \times p_f}.$$

Without loss of generality, we can assume that the common dimensions of $\mathbf{T}$ and $\mathbf{U}$ are the last modes of both tensors. If that is not the case, the modes of the tensors are reordered such that the condition is fulfilled. The contraction of the modes $p_1, \ldots, p_f$ of $\mathbf{T}$ and $\mathbf{U}$ results in a new tensor $\mathbf{V} \in \mathbb{R}^{m_1 \times \cdots \times m_d \times n_1 \times \cdots \times n_e}$ with

$$\mathbf{V}_{x_1,\ldots,x_d,y_1,\ldots,y_e} = \sum_{z_1=1}^{p_1} \cdots \sum_{z_f=1}^{p_f} \mathbf{T}_{x_1,\ldots,x_d,z_1,\ldots,z_f} \cdot \mathbf{U}_{y_1,\ldots,y_e,z_1,\ldots,z_f}.$$

Considering the above operation as a generalization of the inner product of two vectors, we write $\mathbf{V} = \langle \mathbf{T}, \mathbf{U} \rangle_{p_1,\ldots,p_f}$.

2.2.3. Tensor Multiplication

The multiplication of tensors is a specific form of index contraction. It can be seen as a generalization of the standard matrix-by-matrix product.

Definition 2.2.3. *For tensors* $\mathbf{G} \in \mathbb{R}^{M\times N}$ *and* $\mathbf{H} \in \mathbb{R}^{N\times P}$ *with index sets* $M = (m_1,\dots,m_d)^T$, $N = (n_1,\dots,n_d)^T$, *and* $P = (p_1,\dots,p_d)^T$, *the product* $\mathbf{G}\cdot\mathbf{H} \in \mathbb{R}^{M\times P}$ *is defined as*

$$(\mathbf{G}\cdot\mathbf{H})_{x_1,y_1,\dots,x_d,y_d} = \sum_{z_1=1}^{n_1}\dots\sum_{z_d=1}^{n_d} \mathbf{G}_{x_1,z_1,\dots,x_d,z_d}\cdot\mathbf{H}_{z_1,y_1,\dots,z_d,y_d}, \tag{2.2.3}$$

for $1 \le x_i \le m_i$ *and* $1 \le y_i \le p_i$, $i = 1,\dots,d$.

As described in (2.1.5), we can associate any tensor $\mathbf{T} \in \mathbb{R}^N$ of order d with a tensor $\overline{\mathbf{T}} \in \mathbb{R}^{N\times\mathbb{1}}$, $\mathbb{1} = (1,\dots,1)^T$. In this way, we define two special cases related to Definition 2.2.3, namely the right multiplication of an operator $\mathbf{G} \in \mathbb{R}^{M\times N}$ with a tensor $\mathbf{T} \in \mathbb{R}^N$ and the left multiplication with the transpose (2.1.7) of a tensor $\mathbf{T} \in \mathbb{R}^M$. The first can be seen as the counterpart of multiplying a matrix and a column vector, the second as the counterpart of multiplying a row vector and a matrix. For the right multiplication of $\mathbf{G}$ with a tensor $\mathbf{T} \in \mathbb{R}^N$, see (2.1.4), we obtain $\mathbf{G}\cdot\mathbf{T} \in \mathbb{R}^M$, where

$$(\mathbf{G}\cdot\mathbf{T})_{x_1,\dots,x_d} = \left(\mathbf{G}\cdot\overline{\mathbf{T}}\right)_{x_1,1,\dots,x_d,1} = \sum_{y_1=1}^{n_1}\cdots\sum_{y_d=1}^{n_d} \mathbf{G}_{x_1,y_1,\dots,x_d,y_d}\cdot\overline{\mathbf{T}}_{y_1,1,\dots,y_d,1},$$

with $\overline{\mathbf{T}}$ as mentioned before. The left multiplication of an operator $\mathbf{G} \in \mathbb{R}^{M\times N}$ with the transpose of $\mathbf{T} \in \mathbb{R}^M$ is given by $\mathbf{T}^T\cdot\mathbf{G} \in \mathbb{R}^{\mathbb{1}\times N}$, where

$$\left(\mathbf{T}^T\cdot\mathbf{G}\right)_{1,y_1,\dots,1,y_d} = \sum_{x_1=1}^{m_1}\cdots\sum_{x_d=1}^{m_d} \mathbf{T}^T_{1,x_1,\dots,1,x_d}\cdot\mathbf{G}_{x_1,y_1,\dots,x_d,y_d}, \tag{2.2.4}$$

for $1 \le y_i \le n_i$, $i = 1,\dots,d$. The result of (2.2.4) is then the transpose of $\mathbf{G}^T\cdot\mathbf{T}$ which is shown by the following theorem.

Lemma 2.2.4. *For operators* $\mathbf{G},\mathbf{H} \in \mathbb{R}^{M\times N}$ *and a scalar* $\lambda \in \mathbb{R}$, *we obtain the following properties:*

$$\begin{aligned} &\text{(i)} && \left(\mathbf{G}^T\right)^T = \mathbf{G},\\ &\text{(ii)} && (\mathbf{G}+\mathbf{H})^T = \mathbf{G}^T + \mathbf{H}^T,\\ &\text{(iii)} && (\lambda\cdot\mathbf{G})^T = \lambda\cdot\mathbf{G}^T. \end{aligned}$$

For $\mathbf{G} \in \mathbb{R}^{M\times N}$ *and* $\mathbf{H} \in \mathbb{R}^{N\times P}$ *it holds that*

$$\text{(iv)} \quad (\mathbf{G}\cdot\mathbf{H})^T = \mathbf{H}^T\cdot\mathbf{G}^T.$$

Proof. Properties (i), (ii), and (iii) follow directly from the definition of the transpose (2.1.7) and Definitions 2.2.1 and 2.2.2. For property (iv), consider

$$\begin{aligned}(\mathbf{G}\cdot\mathbf{H})^T_{x_1,y_1,\dots,x_d,y_d} &= (\mathbf{G}\cdot\mathbf{H})_{y_1,x_1,\dots,y_d,x_d}\\ &= \sum_{z_1=1}^{n_1}\cdots\sum_{z_d=1}^{n_d}\mathbf{G}_{y_1,z_1,\dots,y_d,z_d}\cdot\mathbf{H}_{z_1,x_1,\dots,z_d,x_d}\\ &= \sum_{z_1=1}^{n_1}\cdots\sum_{z_d=1}^{n_d}\mathbf{H}^T_{x_1,z_1,\dots,x_d,z_d}\cdot\mathbf{G}^T_{z_1,y_1,\dots,z_d,y_d}\\ &= \left(\mathbf{H}^T\cdot\mathbf{G}^T\right)_{x_1,y_1,\dots,x_d,y_d},\end{aligned}$$

with $1 \le x_i \le p_i$ and $1 \le y_i \le m_i$ for $i = 1,\dots,d$. □

2.2.4. Tensor Product

In order to mitigate the curse of dimensionality, we will rely on low-parametric representations of tensors. For an understanding of the different tensor formats in the next chapter, let us recall the definition of the *outer product*. For two vectors $v \in \mathbb{R}^m$ and $w \in \mathbb{R}^n$, the outer product $v \otimes w \in \mathbb{R}^{m\times n}$ corresponds to the dyadic product introduced in [25]:

$$v \otimes w = v \cdot w^T.$$

Thus, the outer product of two vectors defines a matrix with

$$(v \otimes w)_{i,j} = v_i \cdot w_j.$$

The outer product is a special case of the more general *tensor product* which may be applied to tensors with arbitrary numbers of modes.

Definition 2.2.5. *The* tensor product *of two tensors* $\mathbf{T} \in \mathbb{R}^{m_1\times\dots\times m_d}$ *and* $\mathbf{U} \in \mathbb{R}^{n_1\times\dots\times n_e}$ *defines a tensor* $\mathbf{T}\otimes\mathbf{U}$ *with*

$$(\mathbf{T}\otimes\mathbf{U})_{x_1,\dots,x_d,y_1,\dots,y_e} = \mathbf{T}_{x_1,\dots,x_d}\cdot\mathbf{U}_{y_1,\dots,y_e},$$

where $1 \le x_i \le m_i$ *for* $i = 1,\dots,d$ *and* $1 \le y_j \le n_j$ *for* $j = 1,\dots,e$.

In other words, the tensor product of two tensors $\mathbf{T}$ and $\mathbf{U}$ defines a tensor of order $d+e$ that contains all modes m_1 to m_d and n_1 to n_d. The tensor product is a bilinear map, i.e. if we fix one of the tensors we get a linear map on the space

where the other tensor lives such that the following conditions are satisfied

$$\begin{aligned}
&\text{(i)} && (\mathbf{T}_1 + \mathbf{T}_2) \otimes \mathbf{U} = \mathbf{T}_1 \otimes \mathbf{U} + \mathbf{T}_2 \otimes \mathbf{U}, \\
&\text{(ii)} && \mathbf{T} \otimes (\mathbf{U}_1 + \mathbf{U}_2) = \mathbf{T} \otimes \mathbf{U}_1 + \mathbf{T} \otimes \mathbf{U}_2, \\
&\text{(iii)} && (\lambda \cdot \mathbf{T}) \otimes \mathbf{U} = \mathbf{T} \otimes (\lambda \cdot \mathbf{U}) = \lambda \cdot (\mathbf{T} \otimes \mathbf{U}),
\end{aligned}$$

for $\mathbf{T}, \mathbf{T}_1, \mathbf{T}_2 \in \mathbb{R}^{m_1 \times \ldots \times m_d}$, $\mathbf{U}, \mathbf{U}_1, \mathbf{U}_2 \in \mathbb{R}^{n_1 \times \ldots \times n_e}$, and $\lambda \in \mathbb{R}$. The tensor product is associative but non-commutative.

As the following theorem will show, the multiplication of tensor products acts elementwise, i.e. multiplying two tensor products with appropriate modes yields the same result as multiplying the corresponding tensors and then computing the tensor product.

Theorem 2.2.6. *Let* $\mathbf{G} \in \mathbb{R}^{M \times N}$ *and* $\mathbf{H} \in \mathbb{R}^{N \times P}$ *with* $\mathbf{G} = \mathbf{G}_1 \otimes \mathbf{G}_2$ *and* $\mathbf{H} = \mathbf{H}_1 \otimes \mathbf{H}_2$*, where*

$$\begin{aligned}
\mathbf{G}_1 &\in \mathbb{R}^{(m_1 \times n_1) \times \ldots \times (m_e \times n_e)}, \quad \mathbf{G}_2 \in \mathbb{R}^{(m_{e+1} \times n_{e+1}) \times \ldots \times (m_d \times n_d)}, \\
\mathbf{H}_1 &\in \mathbb{R}^{(n_1 \times p_1) \times \ldots \times (n_e \times p_e)}, \quad \mathbf{H}_2 \in \mathbb{R}^{(n_{e+1} \times p_{e+1}) \times \ldots \times (n_d \times p_d)}.
\end{aligned}$$

Then, the product of $\mathbf{G}$ *and* $\mathbf{H}$ *is given by*

$$\mathbf{G} \cdot \mathbf{H} = (\mathbf{G}_1 \otimes \mathbf{G}_2) \cdot (\mathbf{H}_1 \otimes \mathbf{H}_2) = (\mathbf{G}_1 \cdot \mathbf{H}_1) \otimes (\mathbf{G}_2 \cdot \mathbf{H}_2)\,.$$

Proof. For $1 \leq x_i \leq m_i$ and $1 \leq y_i \leq p_i$, $i = 1, \ldots, d$, we obtain

$$\begin{aligned}
(\mathbf{G} \cdot \mathbf{U})_{x_1, y_1, \ldots, x_d, y_d} &= \sum_{z_1=1}^{n_1} \cdots \sum_{z_d=1}^{n_d} \mathbf{G}_{x_1, z_1, \ldots, x_d, z_d} \cdot \mathbf{H}_{z_1, y_1, \ldots, z_d, y_d} \\
&= \sum_{z_1=1}^{n_1} \cdots \sum_{z_d=1}^{n_d} (\mathbf{G}_1)_{x_1, z_1, \ldots, x_e, z_e} \cdot (\mathbf{G}_2)_{x_{e+1}, z_{e+1}, \ldots, x_d, z_d} \\
&\qquad \cdot (\mathbf{H}_1)_{z_1, y_1, \ldots, z_e, y_e} \cdot (\mathbf{T}_2)_{z_{e+1}, y_{e+1}, \ldots, z_d, y_d} \\
&= (\mathbf{G}_1 \cdot \mathbf{H}_1)_{x_1, y_1, \ldots, x_e, y_e} \cdot (\mathbf{G}_2 \cdot \mathbf{H}_2)_{x_{e+1}, y_{e+1}, \ldots, x_d, y_d} \\
&= ((\mathbf{G}_1 \cdot \mathbf{H}_1) \otimes (\mathbf{G}_2 \cdot \mathbf{H}_2))_{x_1, y_1, \ldots, x_d, y_d}\,.
\end{aligned}$$

□

2.3. Graphical Representation

When working with high-dimensional tensors, precise descriptions of tensor operations might be confusing or unclear because of the large number of indices involved in the equations. Therefore, it is worthwhile to make use of a diagrammatic notation in order to visualize calculations. For this representation, which is motivated

by [37], we depict a tensor $\mathbf{T} \in \mathbb{R}^N$, $N = (n_1, \ldots, n_d)^T$, as a circle with d arms indicating the set of modes $n_1, \ldots, n_d$. Figure 2.2 shows some examples.

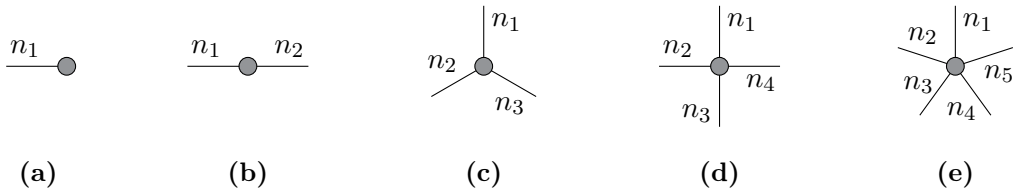


Figure 2.2: *Graphical representation of tensors: (a) Tensor of order 1 (vector). (b) Tensor of order 2 (matrix). (c) Tensor of order 3. (d) Tensor of order 4. (e) Tensor of order 5.*

It is also possible to visualize tensor networks, i.e. couplings of several tensors, in this way. An index contraction of two or more tensors is represented by connecting corresponding arms, see Figure 2.3.

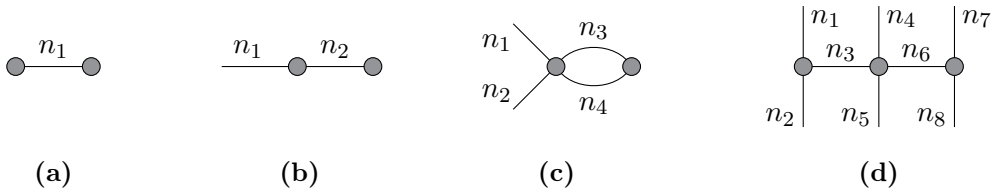


Figure 2.3: *Graphical representation of tensor contractions: (a) Inner product of two vectors. (b) Matrix-vector product. (c) Two-dimensional contraction of two tensors. (d) Contraction of three tensors.*

Figure 2.3 (a) and (b) show the inner product of two vectors and a matrix-vector multiplication, respectively. The contractions depicted in Figure 2.3 (c) and (d) correspond to the equations

$$\mathbf{T}_{x_1,x_2} = \sum_{x_3=1}^{n_3} \sum_{x_4=1}^{n_4} (\mathbf{T}_1)_{x_1,x_2,x_3,x_4} \cdot (\mathbf{T}_2)_{x_3,x_4} \,,$$

and

$$\mathbf{T}_{x_1,x_2,x_4,x_5,x_7,x_8} = \sum_{x_3=1}^{n_3} \sum_{x_6=1}^{n_6} (\mathbf{T}_1)_{x_1,x_2,x_3} \cdot (\mathbf{T}_2)_{x_3,x_4,x_5,x_6} \cdot (\mathbf{T}_3)_{x_6,x_7,x_8} \,,$$

respectively. We will deploy the graphical notation to describe linear tensor operations and tensor algorithms. If it is clear which modes are depicted by all arms, we will omit the labeling.

2.4. Matricization and Vectorization

To describe *matricizations* and *vectorizations* – also called *tensor unfoldings* [37] – we first define a bijection ϕ_N for the index set $N = (n_1, \ldots, n_d)^T \in \mathbb{N}^d$ with

$$\begin{aligned} \phi_N : \{1, \ldots, n_1\} \times \cdots \times \{1, \ldots, n_d\} &\to \{1, \ldots, \prod_{k=1}^{d} n_k\}, \\ (x_1, \ldots, x_d) &\mapsto \phi_N(x_1, \ldots, x_d). \end{aligned}$$

Using the *little-endian convention*, the bijection is defined as

$$\begin{aligned} \phi_N(x_1, \ldots, x_d) &= 1 + (x_1 - 1) + \ldots + (x_d - 1) \cdot n_1 \cdot \ldots \cdot n_{d-1} \\ &= 1 + \sum_{k=1}^{d} (x_k - 1) \prod_{l=1}^{k-1} n_l. \end{aligned} \tag{2.4.1}$$

Of course, one can choose any possible bijection between $\{1, \ldots, n_1\} \times \ldots \times \{1, \ldots, n_d\}$ and $\{1, \ldots, \prod_{k=1}^{d} n_k\}$. However, henceforth we will assume ϕ_N to be defined as above. If the definition of N is clear from the context, we use the multi-index notation

$$\overline{x_1, \ldots, x_d} = \phi_N(x_1, \ldots, x_d). \tag{2.4.2}$$

Theorem 2.4.1. *The function ϕ_N is bijective with inverse ϕ_N^{-1} given by $\phi_N^{-1}(\overline{x_1, \ldots, x_d}) = (\varphi_1, \ldots, \varphi_d)$ with*

$$\varphi_i = \left\lfloor \frac{\overline{x_1, \ldots, x_d} - 1 - \sum_{k=i+1}^{d} (\varphi_k - 1) \prod_{l=1}^{k-1} n_l}{\prod_{l=1}^{i-1} n_l} \right\rfloor + 1,$$

for $1 \leq i \leq d-1$. It holds that $\varphi_i = x_i$.

The proof of Lemma 2.4.1 can be found in Appendix A.1.1. Now, let $N' = (n_{k_1}, \ldots, n_{k_e})^T$ and $N'' = (n_{l_1}, \ldots, n_{l_f})^T$, $d = e + f$, denote two ordered subsets of $N = (n_1, \ldots, n_d)^T$ with properties

$$\begin{aligned} &\text{(i)} \quad N' \cup N'' = N, \\ &\text{(ii)} \quad N' \cap N'' = \varnothing, \end{aligned} \tag{2.4.3}$$

where property (i) does not take the orderings of N, N', and N'' into account. With the aid of the bijection given in (2.4.1), we now define matricizations and vectorizations of tensors.

Definition 2.4.2. *Let $N = (n_1, \dots, n_d)^T$ be an index set and $\mathbf{T} \in \mathbb{R}^N$ a tensor. For two ordered subsets $N' = (n_{k_1}, \dots, n_{k_e})^T$ and $N'' = (n_{l_1}, \dots, n_{l_f})^T$ of N which satisfy* (2.4.3)*, the matricization of $\mathbf{T}$ with respect respect to N' and N'' is given by*

$$\left(\mathbf{T}\middle|\begin{matrix}N''\\N'\end{matrix}\right)_{\overline{x_{k_1},\dots,x_{k_e}},\overline{x_{l_1},\dots,x_{l_f}}} = \mathbf{T}_{x_1,\dots,x_d}. \tag{2.4.4}$$

If we consider an operator $\mathbf{G} \in \mathbb{R}^{M\times N} = \mathbb{R}^{(m_1\times n_1)\times\dots\times(m_d\times n_d)}$ and its "natural" matricization, we write

$$\operatorname{mat}(\mathbf{G}) = \mathbf{G}\left|\begin{matrix}N\\M\end{matrix}\right.. \tag{2.4.5}$$

A vectorization of a tensor $\mathbf{T} \in \mathbb{R}^N$ is given by a matricization of $\mathbf{T}$ where we define $N' = N$ (or a reordering of N) and $N'' = \varnothing$.

Definition 2.4.3. *Let $N = (n_1, \dots, n_d)^T$ be an index set and $\mathbf{T} \in \mathbb{R}^N$ a tensor. For a reordering $N' = (n_{k_1}, \dots, n_{k_d})^T$, the vectorization of $\mathbf{T}$ is given by*

$$\left(\mathbf{T}\middle|_{N'}\right)_{\overline{x_{k_1},\dots,x_{k_d}}} = \mathbf{T}_{x_1,\dots,x_d}.$$

Analogously to (2.4.5), if we consider a tensor $\mathbf{T} \in \mathbb{R}^N$ and its "natural" vectorization, we write

$$\operatorname{vec}(\mathbf{T}) = \mathbf{T}\Big|_N. \tag{2.4.6}$$

Furthermore, for the tensor $\overline{\mathbf{T}} \in \mathbb{R}^{N\times\mathbb{1}}$ as defined in (2.1.5), it holds that

$$\operatorname{mat}(\overline{\mathbf{T}}) = \overline{\mathbf{T}}\left|\begin{matrix}\mathbb{1}\\N\end{matrix}\right. = \mathbf{T}\Big|_N = \operatorname{vec}(\mathbf{T}).$$

Unless otherwise specified, we henceforth regard a vectorization of a tensor $\mathbf{T} \in \mathbb{R}^N$ as a column vector. Matricizations and vectorizations are consistent with the scalar multiplication and addition of tensors as the following lemma shows.

Lemma 2.4.4. *Let $\lambda \in \mathbb{R}$ and $\mathbf{G}, \mathbf{H} \in \mathbb{R}^{M\times N}$ be two tensor operators with index sets M and N. For the scalar multiplication $\lambda \cdot \mathbf{T}$ as defined in* (2.2.2)*, it holds that*

$$\operatorname{mat}(\lambda \cdot \mathbf{T}) = \lambda \cdot \operatorname{mat}(\mathbf{T}).$$

For the addition $\mathbf{T} + \mathbf{U}$ *as defined in (2.2.1), it holds that*

$$\operatorname{mat}(\mathbf{T} + \mathbf{U}) = \operatorname{mat}(\mathbf{T}) + \operatorname{mat}(\mathbf{U}).$$

Proof. The first assertion is clear since multiplication by a scalar is performed elementwise on the entries of $\mathbf{T}$. Following from the definition of matricizations, we obtain

$$\begin{aligned}(\operatorname{mat}(\mathbf{T} + \mathbf{U}))_{\overline{x_1,\ldots,x_d},\overline{y_1,\ldots,y_d}} &= (\mathbf{T} + \mathbf{U})_{x_1,y_1,\ldots,x_d,y_d} \\ &= \mathbf{T}_{x_1,y_1,\ldots,x_d,y_d} + \mathbf{U}_{x_1,y_1,\ldots,x_d,y_d} \\ &= (\operatorname{mat}(\mathbf{T}))_{\overline{x_1,\ldots,x_d},\overline{y_1,\ldots,y_d}} + (\operatorname{mat}(\mathbf{U}))_{\overline{x_1,\ldots,x_d},\overline{y_1,\ldots,y_d}},\end{aligned}$$

for $1 \leq x_i \leq m_i$ and $1 \leq y_i \leq n_i$, $i = 1, \ldots, d$. □

The product of two operator matricizations also corresponds to the matricization of their tensor product.

Theorem 2.4.5. *Let* $\mathbf{G} \in \mathbb{R}^{M \times N}$ *and* $\mathbf{H} \in \mathbb{R}^{N \times P}$ *be two tensor operators with index sets* $M, N, P \in \mathbb{N}^d$*. For the product* $\mathbf{G} \cdot \mathbf{H}$ *as defined in* (2.2.3)*, it holds that*

$$\operatorname{mat}(\mathbf{G} \cdot \mathbf{H}) = \operatorname{mat}(\mathbf{G}) \cdot \operatorname{mat}(\mathbf{H}). \tag{2.4.7}$$

Proof. The assertion follows from

$$\begin{aligned}(\operatorname{mat}(\mathbf{G} \cdot \mathbf{H}))_{\overline{x_1,\ldots,x_d},\overline{y_1,\ldots,y_d}} &= (\mathbf{G} \cdot \mathbf{H})_{x_1,y_1,\ldots,x_d,y_d} \\ &= \sum_{z_1=1}^{n_1} \cdots \sum_{z_d=1}^{n_d} \mathbf{G}_{x_1,z_1,\ldots,x_d,z_d} \cdot \mathbf{H}_{z_1,y_1,\ldots,z_d,y_d} \\ &= \sum_{z=1}^{n_1 \cdot \ldots \cdot n_d} (\operatorname{mat}(\mathbf{G}))_{\overline{x_1,\ldots,x_d},z} \cdot (\operatorname{mat}(\mathbf{H}))_{z,\overline{y_1,\ldots,y_d}},\end{aligned}$$

for $1 \leq x_i \leq m_i$ and $1 \leq y_i \leq p_i$, $i = 1, \ldots, d$. □

If $P = (1, \ldots, 1)^T$ in Theorem 2.4.5, we can identify $\mathbf{H}$ with a tensor $\mathbf{T} \in \mathbb{R}^N$ and (2.4.7) becomes $\operatorname{vec}(\mathbf{G} \cdot \mathbf{T}) = \operatorname{mat}(\mathbf{G}) \cdot \operatorname{vec}(\mathbf{T})$.

2.5. Norms

Similar to classical linear algebra, we define p-norms for tensors. In order to avoid confusion, we will distinguish between norms for tensors in $\mathbb{R}^N$ and tensor operators in $\mathbb{R}^{M \times N}$.

Definition 2.5.1. *For any real number $p \geq 1$, the p-norm of a tensor $\mathbf{T} \in \mathbb{R}^N$ is defined as*

$$\|\mathbf{T}\|_p = \left(\sum_{x_1=1}^{n_1} \cdots \sum_{x_d=1}^{n_d} (\mathbf{T}_{x_1,\ldots,x_d})^p \right)^{1/p}. \tag{2.5.1}$$

We will make no distinction in the notation for p-norms of tensors and p-norms of vectors, because it holds that

$$\|\mathbf{T}\|_p = \|\mathrm{vec}(\mathbf{T})\|_p, \tag{2.5.2}$$

where $\mathrm{vec}(\mathbf{T})$ denotes the "natural" vectorization of $\mathbf{T}$, see (2.4.6). We are aware of several publications, e.g. [42, 63], where the norm (2.5.1) for $p = 2$ is called *Frobenius norm*. However, since we distinguish between tensors in $\mathbb{R}^N$ and $\mathbb{R}^{M\times N}$, we also distinguish between 2-norms of tensors $\mathbf{T} \in \mathbb{R}^N$ and Frobenius norms of tensors $\mathbf{G} \in \mathbb{R}^{M\times N}$. The 2-norm of $\mathbf{T} \in \mathbb{R}^N$ can be expressed using the definition of the transpose of $\mathbf{T}$ given in (2.1.7), i.e.

$$\|\mathbf{T}\|_2 = \sqrt{\mathbf{T}^T \cdot \mathbf{T}}. \tag{2.5.3}$$

Due to the relation given in (2.5.2), we know that $\|\,.\,\|_p$ as a function from $\mathbb{R}^N$ to $\mathbb{R}$ is indeed a norm, i.e. it has the following properties for all $\lambda \in \mathbb{R}$ and all $\mathbf{T}, \mathbf{U} \in \mathbb{R}^N$:

$$\begin{array}{ll} \text{(i)} & \|\lambda \cdot \mathbf{T}\|_p = |\lambda| \cdot \|\mathbf{T}\|_p, \\ \text{(ii)} & \|\mathbf{T} + \mathbf{U}\|_p \leq \|\mathbf{T}\|_p + \|\mathbf{U}\|_p, \\ \text{(iii)} & \|\mathbf{T}\|_p = 0 \Rightarrow \mathbf{T} = 0 \in \mathbb{R}^N. \end{array}$$

Equation (2.5.2) also implies that $\mathbb{R}^N$ is a *Banach space* since the vectorized counterpart of any *Cauchy sequence* $(\mathbf{T}_k)_{k\in\mathbb{N}}$ in $\mathbb{R}^N$ converges in $\mathbb{R}^{n_1\cdot\ldots\cdot n_d}$. Considering operators $\mathbf{G} \in \mathbb{R}^{M\times N}$, we adopt the Frobenius norm for matrices and define

$$\|\mathbf{G}\|_F = \sqrt{\sum_{x_1=1}^{m_1} \sum_{y_1=1}^{n_1} \cdots \sum_{x_d=1}^{m_d} \sum_{y_d=1}^{n_d} (\mathbf{G}_{x_1,y_1,\ldots,x_d,y_d})^2}. \tag{2.5.4}$$

Analogously to (2.5.2), we again have a simple connection to the classical counterpart of (2.5.4):

$$\|\mathbf{G}\|_F = \|\mathrm{mat}(\mathbf{G})\|_F,$$

where mat($\mathbf{G}$) denotes the "natural" matricization of $\mathbf{G}$, see (2.4.5).

Theorem 2.5.2. *The Frobenius norm for tensors is sub-multiplicative and compatible with the 2-norm, i.e. for* $\mathbf{G} \in \mathbb{R}^{M \times N}$, $\mathbf{H} \in \mathbb{R}^{N \times P}$, *and* $\mathbf{T} \in \mathbb{R}^{N}$, *we obtain*

$$\|\mathbf{G} \cdot \mathbf{H}\|_F \leq \|\mathbf{G}\|_F \cdot \|\mathbf{H}\|_F \quad \textit{and} \quad \|\mathbf{G} \cdot \mathbf{T}\|_2 \leq \|\mathbf{G}\|_F \cdot \|\mathbf{T}\|_2 .$$

Proof. We show that $\| \, . \, \|_F$ is sub-multiplicative, the proof for compatibility with the 2-norm is analogous. Because the classical Frobenius norm for matrices is sub-multiplicative, it holds that

$$\begin{aligned} \|\mathbf{G} \cdot \mathbf{H}\|_F &= \|\mathrm{mat}\,(\mathbf{G}) \cdot \mathrm{mat}\,(\mathbf{H})\|_F \\ &\leq \|\mathrm{mat}\,(\mathbf{G})\|_F \cdot \|\mathrm{mat}\,(\mathbf{H})\|_F \\ &= \|\mathbf{G}\|_F \cdot \|\mathbf{H}\|_F . \end{aligned}$$

□

2.6. Orthonormality

The last section of this chapter generalizes the concept of orthonormal matrices to tensors of higher order. Henceforth, we will call a matrix $A \in \mathbb{R}^{m \times n}$ *orthonormal with respect to the rows* if

$$A \cdot A^T = I \in \mathbb{R}^{m \times m}.$$

Analogously, we call a matrix $A \in \mathbb{R}^{m \times n}$ *orthonormal with respect to the columns* if

$$A^T \cdot A = I \in \mathbb{R}^{n \times n}.$$

That is, a matrix is orthonormal with respect to the rows (columns) if the rows (columns) form an orthonormal set. This basic concept can be adapted to tensors in the following way.

Definition 2.6.1. *Let* $\mathbf{T} \in \mathbb{R}^{N}$, $N = (n_1, \ldots, n_d)^T$, *be a tensor and* $N', N'' \subset N$ *a splitting of the modes with* $N' = (n_{k_1}, \ldots, n_{k_e})^T$ *and* $N'' = (n_{l_1}, \ldots, n_{l_f})^T$, $e+f = d$. $\mathbf{T}$ *is called orthonormal with respect to* N' *if the matricization of* $\mathbf{T}$ *with respect to the sets* N' *and* N'' (2.4.4) *satisfies*

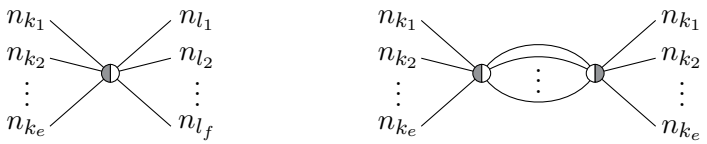
$$\mathbf{T}\Big|_{N'}^{N''} \cdot \left(\mathbf{T}\Big|_{N'}^{N''}\right)^T = \mathbf{T}\Big|_{N'}^{N''} \cdot \mathbf{T}\Big|_{N''}^{N'} = I \in \mathbb{R}^{N' \times N'}.$$

Note that a reordering within the sets N' and N'' would not affect the orthonor-

mality of the tensor $\mathbf{T}$. Furthermore, it is necessary that

$$\prod_{i=1}^{e} n_{k_i} \leq \prod_{i=1}^{f} n_{l_i},$$

since the rows of $\mathbf{T}\Big|_{N'}^{N''}$ have to form an orthonormal set. In order to visualize orthonormal tensors we make use of the graphical representation described in Section 2.3. We draw half filled circles indicating the orthonormality, cf. [37]. Figure 2.4 shows the notation for a tensor $\mathbf{T} \in \mathbb{R}^N$ and sets $N', N'' \subset N$ as above.



(a) **(b)**

Figure 2.4: *Orthonormal tensors: (a) Graphical representation of a tensor $\mathbf{T} \in \mathbb{R}^N$ which is orthonormal with respect to the set $N' = (n_{k_1}, \ldots, n_{k_e})^T \subset N$. (b) Tensor multiplication of $\mathbf{T}$ and $\mathbf{T}^T$. The result is the identity tensor in $\mathbb{R}^{N' \times N'}$.*

With the definitions above, we can also describe *QR decompositions* and *singular value decompositions* (SVD) for tensors. Just as one can decompose a matrix into a product of an orthonormal matrix and an upper triangular matrix, it is possible to perform a QR decomposition (or SVD) of a tensor or rather of a matricization of it. As a reminder, any real matrix $A \in \mathbb{R}^{m \times n}$ can be decomposed as

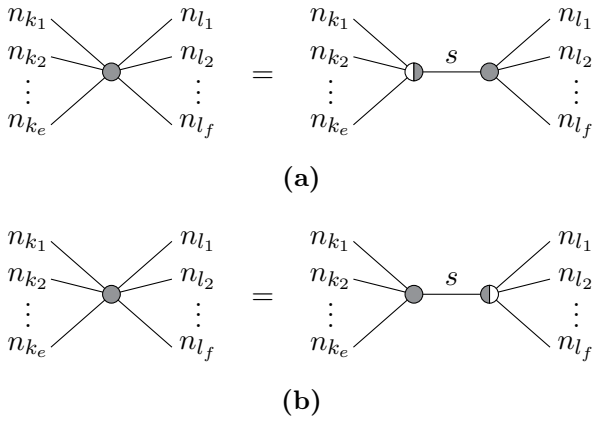
$$A = Q \cdot R,$$

where Q is an orthonormal matrix with respect to its columns, i.e. $Q^T \cdot Q = I$ and R is an upper triangular matrix. For $m \leq n$, we obtain $Q \in \mathbb{R}^{m \times m}$ and $R \in \mathbb{R}^{m \times n}$. If $m > n$, a *reduced QR decomposition* – or *thin QR factorization* [64] – can be computed, i.e. $Q \in \mathbb{R}^{m \times n}$ and $R \in \mathbb{R}^{n \times n}$.

Furthermore, $A \in \mathbb{R}^{m \times n}$ can be decomposed as

$$A = U \cdot \Sigma \cdot V^T,$$

where U and V are orthonormal with respect to their columns and Σ is a diagonal matrix containing the singular values of A. If we consider a full SVD, U and V are also orthonormal with respect to their rows and Σ may contain diagonal entries equal to zero. However, in what follows, we will require that only the non-zero singular values $\sigma_1, \ldots, \sigma_s$, $s \leq \min\{m, n\}$, are stored in the diagonal of Σ resulting in a *compact SVD*. Additionally, we assume that these singular values are sorted in decreasing order, i.e. $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_s > 0$.



(a)

(b)

Figure 2.5: *QR decompositions of a tensor: (a) QR decomposition corresponding to* (2.6.1)*. (b) QR decomposition corresponding to* (2.6.2)*.*

Given a tensor $\mathbf{T} \in \mathbb{R}^N$ and a mode splitting $N', N'' \subset N$ with sets $N' = (n_{k_1}, \ldots, n_{k_e})^T$ and $N'' = (n_{l_1}, \ldots, n_{l_f})^T$, we can compute two different kinds of QR decompositions by either computing a QR factorization of $\mathbf{T}\Big|_{N'}^{N''}$ or $\mathbf{T}\Big|_{N''}^{N'}$, respectively. In these cases, we obtain

$$\mathbf{T}\Big|_{N'}^{N''} = Q \cdot R = \mathbf{Q}\Big|_{N'}^{s} \cdot \mathbf{R}\Big|_{s}^{N'}, \tag{2.6.1}$$

and

$$\left(\mathbf{T}\Big|_{N'}^{N''}\right)^T = Q \cdot R = \mathbf{Q}\Big|_{N''}^{s} \cdot \mathbf{R}\Big|_{s}^{N'} \quad \Leftrightarrow \quad \mathbf{T}\Big|_{N'}^{N''} = \mathbf{R}\Big|_{N'}^{s} \cdot \mathbf{Q}\Big|_{s}^{N'}, \tag{2.6.2}$$

respectively. It holds that $s \leq \min\{\prod_{i=1}^{e} n_{k_i}, \prod_{i=1}^{f} n_{l_i}\}$. Figure 2.5 shows the graphical notation of these decompositions after reshaping the matricizations of $\mathbf{Q}$ and $\mathbf{R}$. If we compute an SVD of $\mathbf{T}\Big|_{N'}^{N''}$, the resulting tensor network is orthonormal on both of its sides, see Figure 2.6.

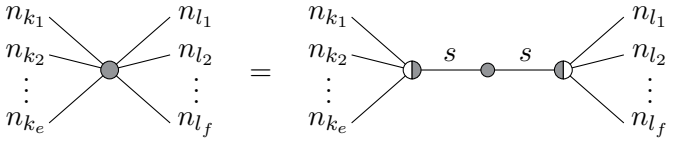


Figure 2.6: *Singular value decomposition of a tensor: The matricization of* $\mathbf{T}$ *with respect to the mode sets* N' *and* N'' *is decomposed into two orthonormal tensors and a matrix containing the singular values (depicted by a small circle).*

3
Tensor Decomposition

In this chapter, we will describe tensor approximations and decompositions in detail. Over the last decades, various tensor formats have been developed. The common basis of these formats is the tensor product which enables us to decompose high-dimensional tensors into several smaller tensors. Here, we will focus on the so-called tensor-train format, which is a promising candidate for approximating high-dimensional tensors by low-rank decompositions. After introducing rank-one tensors, the canonical format, and the (hierarchical) Tucker format, we will explain the basic concept of the tensor-train format and show how to generalize well-known operations of standard matrix and vector calculus. Furthermore, we will describe three formats related to tensor trains, namely the quantized tensor-train format, the block tensor-train format, and the cyclic tensor-train format. Note that, in what follows, we will use the notation of tensors ($\mathbf{T}$, $\mathbf{U}$, $\mathbf{G}$, etc.) for different formats.

3.1. Rank-One Tensors

The very basis of tensor decompositions as presented in this work is the idea to consider high-dimensional tensors that can be represented as tensor products of a set of vectors, so-called *rank-one tensors* [65] or *elementary tensors* [66].

Definition 3.1.1. *A tensor* $\mathbf{T} \in \mathbb{R}^N$, $\mathbb{R}^N = \mathbb{R}^{n_1 \times \cdots \times n_d}$, *of order* d *is called* rank-one tensor *if it can be written as the tensor product of* d *vectors, i.e.*

$$\mathbf{T} = \bigotimes_{i=1}^{d} \mathbf{T}^{(i)} = \mathbf{T}^{(1)} \otimes \cdots \otimes \mathbf{T}^{(d)}, \tag{3.1.1}$$

where $\mathbf{T}^{(i)} \in \mathbb{R}^{n_i}$ *for* $i = 1, \ldots, d$.

Any element of $\mathbf{T}$ is then the product of the corresponding elements of the different vectors $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(d)}$. That is, we can write

$$\mathbf{T}_{x_1,\ldots,x_d} = \prod_{i=1}^{d} \mathbf{T}^{(i)}_{x_i} = \mathbf{T}^{(1)}_{x_1} \cdot \ldots \cdot \mathbf{T}^{(d)}_{x_d}.$$

If we consider linear operators $\mathbf{G} \in \mathbb{R}^{M \times N}$ with $\mathbb{R}^{M \times N} = \mathbb{R}^{(m_1 \times n_1) \times \cdots \times (m_d \times n_d)}$,

that can be expressed as rank-one tensors, the components $\mathbf{G}^{(i)}$ are matrices, i.e.

$$\mathbf{G} = \bigotimes_{i=1}^{d} \mathbf{G}^{(i)} = \mathbf{G}^{(1)} \otimes \cdots \otimes \mathbf{G}^{(d)}, \tag{3.1.2}$$

with $\mathbf{G}^{(i)} \in \mathbb{R}^{m_i \times n_i}$ for $i = 1, \ldots, d$. The storage consumption of rank-one tensors can be estimated as $O(n \cdot d)$ for tensors $\mathbf{T} \in \mathbb{R}^N$ and $O(m \cdot n \cdot d)$ for tensor operators $\mathbf{G} \in \mathbb{R}^{M \times N}$, where m and n are the maximums of all mode sizes given by M and N, respectively. Note that the rank-one decompositions given in (3.1.1) and (3.1.2) are not unique in the sense that we can multiply every component $\mathbf{T}^{(i)}$ (or $\mathbf{G}^{(i)}$) with a scalar value $\lambda_i \in \mathbb{R}$ as long as they satisfy $\prod_{i=1}^{d} \lambda_i = 1$.

As already mentioned in the previous chapter, we can always associate a tensor $\mathbf{T} \in \mathbb{R}^N$ with a tensor $\overline{\mathbf{T}} \in \mathbb{R}^{N \times \mathbb{1}}$, see (2.1.5). For that reason, we will only consider addition and multiplication rules for rank-one operators. The rules then apply to any rank-one decomposition. Let us consider the multiplication of two rank-one tensor operators $\mathbf{G} \in \mathbb{R}^{M \times N}$ and $\mathbf{H} \in \mathbb{R}^{N \times P}$ with index sets M, N, and P. One can see that the product $\mathbf{G} \cdot \mathbf{H} \in \mathbb{R}^{M \times P}$ as defined in (2.2.3) is again a rank-one tensor, since it follows from Theorem 2.2.6 that

$$\mathbf{G} \cdot \mathbf{H} = \left(\bigotimes_{i=1}^{d} \mathbf{G}^{(i)} \right) \cdot \left(\bigotimes_{i=1}^{d} \mathbf{H}^{(i)} \right) = \bigotimes_{i=1}^{d} \left(\mathbf{G}^{(i)} \cdot \mathbf{H}^{(i)} \right), \tag{3.1.3}$$

where $\mathbf{G}^{(i)} \in \mathbb{R}^{m_i \times n_i}$ and $\mathbf{H}^{(i)} \in \mathbb{R}^{n_i \times p_i}$ for $i = 1, \ldots, d$. Contrary to this, the sum of two rank-one tensors might not be a rank-one tensor.

In general, an approximation of a given tensor by a rank-one tensor is rather inaccurate. However, we can extend the basic idea of rank-one tensors for more complex tensor decompositions and approximations as shown in the next sections.

3.2. Canonical Format

The idea of the *canonical format* is to express a tensor as the sum of a finite number of rank-one tensors [33].

Definition 3.2.1. *A tensor $\mathbf{T} \in \mathbb{R}^N$ is said to be in the* canonical format *if*

$$\mathbf{T} = \sum_{k=1}^{r} \bigotimes_{i=1}^{d} \mathbf{T}^{(i)}_{k,:} = \sum_{k=1}^{r} \mathbf{T}^{(1)}_{k,:} \otimes \cdots \otimes \mathbf{T}^{(d)}_{k,:},$$

with cores $\mathbf{T}^{(i)} \in \mathbb{R}^{r \times n_i}$ *for $i = 1, \ldots, d$, where r is called the* canonical rank *of the decomposition.*

Here, we add an additional dimension to the components $\mathbf{T}^{(i)}$. Fixing the first

index, $\mathbf{T}^{(i)}_{k,:} \in \mathbb{R}^{n_i}$ denotes the (transposed) k-th row of the matrix $\mathbf{T}^{(i)}$. Any element of $\mathbf{T}$ can be written as

$$\mathbf{T}_{x_1,\dots,x_d} = \sum_{k=1}^{r} \prod_{i=1}^{d} \mathbf{T}^{(i)}_{k,x_i} = \sum_{k=1}^{r} \mathbf{T}^{(1)}_{k,x_1} \cdot \ldots \cdot \mathbf{T}^{(d)}_{k,x_d}.$$

Definition 3.2.2. *A tensor operator $\mathbf{G} \in \mathbb{R}^{M \times N}$ is said to be in the canonical format if*

$$\mathbf{G} = \sum_{k=1}^{r} \bigotimes_{i=1}^{d} \mathbf{G}^{(i)}_{k,:,:} = \sum_{k=1}^{r} \mathbf{G}^{(1)}_{k,:,:} \otimes \cdots \otimes \mathbf{G}^{(d)}_{k,:,:},$$

with cores $\mathbf{G}^{(i)} \in \mathbb{R}^{r \times m_i \times n_i}$ *for* $i = 1, \dots, d$.

The elements of $\mathbf{G}$ can then be written as

$$\mathbf{G}_{x_1,y_1,\dots,x_d,y_d} = \sum_{k=1}^{r} \prod_{i=1}^{d} \mathbf{G}^{(i)}_{k,x_i,y_i} = \sum_{k=1}^{r} \mathbf{G}^{(1)}_{k,x_1,y_1} \cdot \ldots \cdot \mathbf{G}^{(d)}_{k,x_d,y_d}.$$

In fact, any tensor can be represented as a linear combination of such elementary tensors. The crucial point is the number of required rank-one tensors. If the rank r is small enough, we may reduce the storage consumption of an order-d tensor with the aid of the canonical format significantly. Instead of an exponential dependence as in the full format, the storage only depends linearly on the number of dimensions for tensors in the canonical format.

Lemma 3.2.3. *The storage consumptions of tensors $\mathbf{T} \in \mathbb{R}^N$ and $\mathbf{G} \in \mathbb{R}^{M \times N}$ in the canonical format can be estimated as $O(r_{\mathbf{T}} \cdot n \cdot d)$ and $O(r_{\mathbf{G}} \cdot m \cdot n \cdot d)$, respectively, where $r_{\mathbf{T}}, r_{\mathbf{G}} \in \mathbb{N}$ are the canonical ranks of $\mathbf{T}$ and $\mathbf{G}$. The number m is the maximum of all mode sizes of $M \in \mathbb{N}^d$ and n is the maximum of all mode sizes of $N \in \mathbb{N}^d$.*

Proof. The storage consumption of a core $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{\mathbf{T}} \times n_i}$ is estimated as $O(r_{\mathbf{T}} \cdot n)$ and the storage consumption of a core $\mathbf{G}^{(i)} \in \mathbb{R}^{r_{\mathbf{G}} \times m_i \times n_i}$ is estimated as $O(r_{\mathbf{G}} \cdot m \cdot n)$. Summation over all cores concludes the proof. □

Similar to full tensors and rank-one tensors, we can again associate a tensor $\mathbf{T} \in \mathbb{R}^N$ in canonical format with rank r with a canonical tensor operator $\overline{\mathbf{T}} \in \mathbb{R}^{N \times \mathbb{1}}$, $\mathbb{1} = (1, \dots, 1)^T$, by setting

$$\overline{\mathbf{T}}^{(i)}_{k,x_i,1} = \mathbf{T}^{(i)}_{k,x_i},$$

for $k = 1, \dots, r$, $x_i = 1, \dots, n_i$, and $i = 1, \dots, d$. Note that $\overline{\mathbf{T}}$ also has rank r.

Furthermore, the transpose of a tensor operator $\mathbf{G} \in \mathbb{R}^{M \times N}$ in canonical format is given by

$$\mathbf{G}^T = \sum_{k=1}^{r} \bigotimes_{i=1}^{d} \left(\mathbf{G}_{k,:,:}^{(i)} \right)^T. \tag{3.2.1}$$

Multiplying a canonical tensor with a scalar $\lambda \in \mathbb{R}$ means multiplying every rank-one tensor with λ. Thus, scalar multiplication has no influence on the rank of a tensor in canonical format. When adding two tensors in canonical format, the rank of the resulting tensor is less than or equal to the sum of the ranks of both addends. That is, adding two tensor operators $\mathbf{G}_1 \in \mathbb{R}^{M \times N}$ with canonical rank $r_1 \in \mathbb{N}$ and $\mathbf{G}_2 \in \mathbb{R}^{M \times N}$ with canonical rank $r_2 \in \mathbb{N}$, we obtain in general

$$\mathbf{G} = \mathbf{G}_1 + \mathbf{G}_2 = \sum_{k=1}^{r_1+r_2} \bigotimes_{i=1}^{d} \mathbf{G}_{k,:,:}^{(i)},$$

with

$$\mathbf{G}_{k,:,:}^{(i)} = \begin{cases} \left(\mathbf{G}_1^{(i)} \right)_{k,:,:}, & \text{if} \quad 1 \leq k \leq r_1, \\ \\ \left(\mathbf{G}_2^{(i)} \right)_{k-r_1,:,:}, & \text{if} \quad r_1 + 1 \leq k \leq r_1 + r_2, \end{cases}$$

for $i = 1, \ldots, d$. Considering the multiplication of two tensor operators $\mathbf{G}_1 \in \mathbb{R}^{M \times N}$ and $\mathbf{G}_2 \in \mathbb{R}^{N \times P}$ in the canonical format, the upper bound for the canonical rank of $\mathbf{G}_1 \cdot \mathbf{G}_2$ is equal to the product of the ranks of $\mathbf{G}_1$ and $\mathbf{G}_2$. It follows from (3.1.3) that a canonical representation of the product of $\mathbf{G}_1$ and $\mathbf{G}_2$ is given by

$$\mathbf{G} = \mathbf{G}_1 \cdot \mathbf{G}_2 = \sum_{k_1=1}^{r_1} \sum_{k_2=1}^{r_2} \bigotimes_{i=1}^{d} \left(\left(\mathbf{G}_1^{(i)} \right)_{k_1,:,:} \cdot \left(\mathbf{G}_2^{(i)} \right)_{k_2,:,:} \right). \tag{3.2.2}$$

Using the notation from Section 2.4, we can write (3.2.2) as

$$\mathbf{G} = \sum_{k=1}^{r_1 \cdot r_2} \bigotimes_{i=1}^{d} \mathbf{G}_{k,:,:}^{(i)},$$

with $\mathbf{G}_{\overline{k_1,k_2},:,:}^{(i)} = \left(\mathbf{G}_1^{(i)} \right)_{k_1,:,:} \cdot \left(\mathbf{G}_2^{(i)} \right)_{k_2,:,:}$.

In practice, adding and multiplying canonical tensors can be extremely expensive

in terms of storage consumption and computational cost. Additionally, even if the canonical format can be used for low-parametric decompositions of high-dimensional tensors, it has a crucial drawback. Because canonical tensors with bounded rank r do not form a manifold, robust algorithms for the computation of best approximations are not available. In the canonical format, optimization problems can be ill-posed [36], with the result that the best approximation may not even exist. However, we will make use of the canonical format later, e.g. to describe tensor decompositions of master equations. For more information on working with canonical tensors, we refer to [67].

3.3. Tucker and Hierarchical Tucker Format

Compared to the canonical format, the *Tucker format* [38, 39] provides the advantage of being algorithmically stable, i.e. a best approximation of a given tensor always exists [68]. The basic idea is to decompose a tensor into several factor matrices and a core tensor which has the same order as the given tensor but smaller mode sizes.

Definition 3.3.1. *A tensor* $\mathbf{T} \in \mathbb{R}^N$ *is said to be in the* Tucker format *if*

$$\begin{aligned}\mathbf{T} &= \sum_{k_1=1}^{r_1} \cdots \sum_{k_d=1}^{r_d} \left(\mathbf{T}^{(1)}_{:,k_1} \otimes \cdots \otimes \mathbf{T}^{(d)}_{:,k_d} \right) \cdot \mathbf{U}_{k_1,\ldots,k_d} \\ &= \left(\mathbf{T}^{(1)} \otimes \cdots \otimes \mathbf{T}^{(d)} \right) \cdot \mathbf{U},\end{aligned}$$

where $\mathbf{U} \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ *is called* Tucker core, $\mathbf{T}^{(i)} \in \mathbb{R}^{n_i \times r_i}$, $i = 1, \ldots, d$, *are called* Tucker factors *and the numbers* r_i *are called* Tucker ranks.

The canonical format can be viewed as a special case of the Tucker format with $r_1 = r_2 = \cdots = r_d =: r$ and a core tensor $\mathbf{U} \in \mathbb{R}^{r \times \cdots \times r}$ defined by

$$\mathbf{U}_{k_1,\ldots,k_d} = \delta_{k_1,\ldots,k_d},$$

where $\delta_{k_1,\ldots,k_d}$ is a generalized form of the Kronecker delta with

$$\delta_{k_1,\ldots,k_d} = \begin{cases} 1, & \text{if } k_1 = k_2 = \cdots = k_d, \\ 0, & \text{otherwise.} \end{cases}$$

A graphical representation of a Tucker tensor $\mathbf{T} \in \mathbb{R}^N$ is shown in Figure 3.1 (a).

As mentioned before, unlike the canonical format, the Tucker format does not suffer from the ill-posedness of approximation problems. However, the storage consumption of canonical tensors only depends linearly on the order which is not the case for Tucker decompositions.

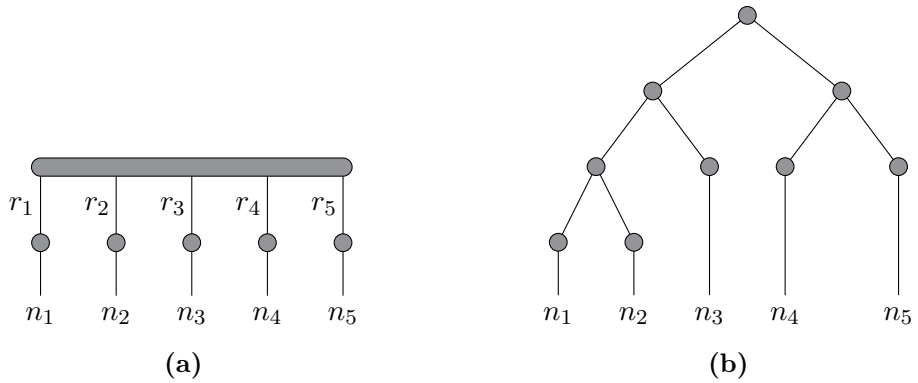


Figure 3.1: *Graphical representation of the Tucker format and the HT format: (a) Tensor of order 5 in Tucker format, the bar depicts the Tucker core and the Tucker factors are represented by the circles. (b) Tensor of order 5 in HT-format, the circles at the inner nodes of the partition tree are transfer tensors, the circles at the leaves are matrices where one dimension corresponds to a mode of the represented tensor.*

Lemma 3.3.2. *The storage consumption of a tensor $\mathbf{T} \in \mathbb{R}^N$ in the Tucker format can be estimated as $O(r \cdot n \cdot d + r^d)$, where $r \in \mathbb{N}$ is the maximum of all Tucker ranks, $n \in \mathbb{N}$ is the maximum of all mode sizes of N, and $d \in \mathbb{N}$ is the order of $\mathbf{T}$.*

Proof. The storage of the d Tucker factors can be estimated as $O(r \cdot n \cdot d)$. Additionally, the Tucker core is a d-dimensional tensor. The number of its entries is bounded by r^d. □

This shows that the curse of dimensionality may not be mitigated by this approach. Due to the exponential scaling in d, we would only benefit from the Tucker format if we assume the ranks to be small enough. Therefore, the idea of the Tucker representation was generalized to tree-structured tensor decompositions, namely the so-called *hierarchical Tucker format* (HT format), see [43, 44, 66]. In this framework, a tensor is partitioned into a dimension tree with matrices at its leaves containing the modes of the tensor. These matrices are linked by the so-called *transfer tensors*, which are located at all inner nodes of the partition tree. An example for the graphical representation of a tensor $\mathbf{T} \in \mathbb{R}^N$ in the HT format is shown in Figure 3.1 (b).

The HT format combines the advantages of the canonical format and the Tucker format. Firstly, there exist robust algorithms to compute best approximations and, secondly, the storage consumption of a tensor in HT format does not depend exponentially on the order d.

Lemma 3.3.3. *The storage consumption of a tensor $\mathbf{T} \in \mathbb{R}^N$ in the HT format can be estimated as $O(r \cdot n \cdot d + r^3 \cdot d)$, where $r \in \mathbb{N}$ is the maximum of all HT ranks, $n \in \mathbb{N}$ is the maximum of all mode sizes of N and $d \in \mathbb{N}$ is the order of $\mathbf{T}$.*

Proof. Following the definition of the HT format, a partition tree is a full binary tree, i.e. a tree in which every node has either zero or two children. In a non-empty full binary tree, the number of leaves and the number of internal nodes differ by one.

That is, there are $d-1$ transfer tensors of order 3 (each mode is bounded by r) and there are d leaves, whose storage consumption can be estimated as $O(r \cdot n \cdot d)$. □

Since we will focus on a special case of the HT format, we refer the interested reader to [15] and [69] for more details about tensor decompositions and computations in the HT format in general.

3.4. Tensor-Train Format

In terms of storage consumption and computational robustness, a promising candidate is the *tensor-train format* (TT format) [16, 17], which combines the main advantages of the canonical format and the Tucker format. Again, the idea is to decompose tensors $\mathbf{T} \in \mathbb{R}^N$ and $\mathbf{G} \in \mathbb{R}^{M \times N}$ into d component tensors $\mathbf{T}^{(i)}$ and $\mathbf{G}^{(i)}$, respectively.

Definition 3.4.1. *A tensor* $\mathbf{T} \in \mathbb{R}^N$ *is said to be in the* TT format *if*

$$\mathbf{T} = \sum_{k_0=1}^{r_0} \cdots \sum_{k_d=1}^{r_d} \bigotimes_{i=1}^{d} \mathbf{T}^{(i)}_{k_{i-1},:,k_i} = \sum_{k_0=1}^{r_0} \cdots \sum_{k_d=1}^{r_d} \mathbf{T}^{(1)}_{k_0,:,k_1} \otimes \cdots \otimes \mathbf{T}^{(d)}_{k_{d-1},:,k_d}.$$

The tensors $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$ *of order 3 are called* TT cores *and the numbers* r_i *are called* TT ranks. *It holds that* $r_0 = r_d = 1$ *and* $r_i \geq 1$ *for* $i = 1, \ldots, d-1$.

Any element of $\mathbf{T}$ can then be written as

$$\mathbf{T}_{x_1,\ldots,x_d} = \sum_{k_0=1}^{r_0} \cdots \sum_{k_d=1}^{r_d} \mathbf{T}^{(1)}_{k_0,x_1,k_1} \cdot \ldots \cdot \mathbf{T}^{(d)}_{k_{d-1},x_d,k_d} = \mathbf{T}^{(1)}_{:,x_1,:} \cdot \ldots \cdot \mathbf{T}^{(d)}_{:,x_d,:}. \tag{3.4.1}$$

Since the right-hand side of (3.4.1) has to be scalar, it is necessary that $r_0 = r_d = 1$. As we will explain later, this condition can be generalized to $r_0 = r_d \geq 1$ yielding a modified version of the TT format, see Section 3.5.3. The TT ranks determine the storage consumption of a tensor train and have a strong influence on the possible complexity, i.e. the capability of representing a given tensor as a tensor train. The lower the ranks, the lower are the memory consumption and computational costs.

Definition 3.4.2. *A tensor operator* $\mathbf{G} \in \mathbb{R}^{M \times N}$ *is said to be in the* TT format *if*

$$\mathbf{G} = \sum_{k_0=1}^{r_0} \cdots \sum_{k_d=1}^{r_d} \bigotimes_{i=1}^{d} \mathbf{G}^{(i)}_{k_{i-1},:,:,k_i} = \sum_{k_0=1}^{r_0} \cdots \sum_{k_d=1}^{r_d} \mathbf{G}^{(1)}_{k_0,:,:,k_1} \otimes \cdots \otimes \mathbf{G}^{(d)}_{k_{d-1},:,:,k_d},$$

with TT cores $\mathbf{G}^{(i)} \in \mathbb{R}^{r_{i-1} \times m_i \times n_i \times r_i}$ *for* $i = 1, \ldots, d$ *and* $r_0 = r_d = 1$.

Again, assuming $r_0 = r_d = 1$, any element of $\mathbf{G}$ can then be written as

$$\mathbf{G}_{x_1,y_1,\ldots,x_d,y_d} = \mathbf{G}^{(1)}_{:,x_1,y_1,:} \cdot \ldots \cdot \mathbf{G}^{(d)}_{:,x_d,y_d,:}.$$

The transpose of a tensor $\mathbf{G} \in \mathbb{R}^{M \times N}$ in the TT format is given by

$$\mathbf{G}^T = \sum_{k_0=1}^{r_0} \cdots \sum_{k_d=1}^{r_d} \bigotimes_{i=1}^{d} \left(\mathbf{G}^{(i)}_{k_{i-1},:,:,k_i} \right)^T,$$

which corresponds to the definitions of the transpose in full format (2.1.6) and canonical format (3.2.1). Figure 3.2 shows the graphical representation of a tensor train $\mathbf{T} \in \mathbb{R}^N$ and a TT operator $\mathbf{G} \in \mathbb{R}^{M \times N}$. As described in Section 2.3, a core is depicted by a circle with different arms indicating the modes of the tensor and the rank indices. We regard the first and the last TT core as matrices due to the fact that $r_0 = r_d = 1$. Analogously, the first and the last core of $\mathbf{G}$ are interpreted as tensors of order 3.

Figure 3.2: *Graphical representation of tensor trains: (a) Tensor of order 5 in TT format, the first and the last core are matrices, the other cores are tensors of order 3. (b) Linear operator of order 10 in TT format, the first and the last core are tensors of order 3, the other cores are tensors of order 4.*

As for the canonical format, the storage consumption for tensors in the TT format only depends linearly on the number of dimensions.

Lemma 3.4.3. *The storage consumptions of tensors $\mathbf{T} \in \mathbb{R}^N$ and $\mathbf{G} \in \mathbb{R}^{M \times N}$ in the TT format can be estimated as $O(r_{\mathbf{T}}^2 \cdot n \cdot d)$ and $O(r_{\mathbf{G}}^2 \cdot m \cdot n \cdot d)$, respectively, where $r_{\mathbf{T}}, r_{\mathbf{G}} \in \mathbb{N}$ are maximums of all TT ranks of $\mathbf{T}$ and $\mathbf{G}$. The number m is the maximum of all mode sizes of $M \in \mathbb{N}^d$ and n is the maximum of all mode sizes of $N \in \mathbb{N}^d$.*

Proof. The storage consumption of each of the d cores $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$ can be estimated as $O(r^2 \cdot n)$. The storage consumption of each of the d cores $\mathbf{G}^{(i)} \in \mathbb{R}^{r_{i-1} \times m_i \times n_i \times r_i}$ can be estimated as $O(r^2 \cdot m \cdot n)$. □

The TT format can be seen as a special case of the hierarchical Tucker format [66], see Figure 3.3. Thus, we benefit from the properties of the HT format, i.e. the non-exponential dependence on the order and the well-posedness of optimization

problems. That is, the main advantages of the TT format, compared to the canonical format, is its stability from an algorithmic point of view. The existence of a best approximation with bounded TT ranks is always ensured [37, 70]. Hence, the TT format format is stable in the sense that we can compute quasi-optimal approximations by sequences of SVDs [17, 69]. With the TT format, we are able to mitigate the curse of dimensionality as long as the ranks and modes are of manageable size.

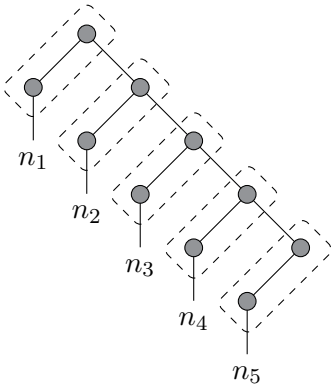


Figure 3.3: *The TT format as a special case of the HT format: At every level, one mode is separated. As indicated by the dashed rectangles, the TT representation is obtained by contracting a matrix at a leaf with the corresponding transfer tensor*

3.4.1. Core Notation

For the sake of comprehensibility, we represent the TT cores as two-dimensional arrays containing vectors or matrices as elements, respectively. For a given tensor train $\mathbf{T} \in \mathbb{R}^N$ with cores $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$, a single core is written as

$$\left[\mathbf{T}^{(i)}\right] = \begin{bmatrix} \mathbf{T}^{(i)}_{1,:,1} & \cdots & \mathbf{T}^{(i)}_{1,:,r_i} \\ \vdots & \ddots & \vdots \\ \mathbf{T}^{(i)}_{r_{i-1},:,1} & \cdots & \mathbf{T}^{(i)}_{r_{i-1},:,r_i} \end{bmatrix}. \tag{3.4.2}$$

For a given operator $\mathbf{G} \in \mathbb{R}^{M \times N}$ with cores $\mathbf{G}^{(i)} \in \mathbb{R}^{R_{i-1} \times m_i \times n_i \times R_i}$, each core is written as

$$\left[\mathbf{G}^{(i)}\right] = \begin{bmatrix} \mathbf{G}^{(i)}_{1,:,:,1} & \cdots & \mathbf{G}^{(i)}_{1,:,:,R_i} \\ \vdots & \ddots & \vdots \\ \mathbf{G}^{(i)}_{R_{i-1},:,:,1} & \cdots & \mathbf{G}^{(i)}_{R_{i-1},:,:,R_i} \end{bmatrix}. \tag{3.4.3}$$

We then use the notations

$$\mathbf{T} = \left[\mathbf{T}^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{T}^{(d)}\right] \quad \text{and} \quad \mathbf{G} = \left[\mathbf{G}^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{G}^{(d)}\right]$$

for tensor trains $\mathbf{T} \in \mathbb{R}^N$ and $\mathbf{G} \in \mathbb{R}^{M \times N}$, respectively, cf. [71]. The corresponding operations can be regarded as a generalization of the standard matrix multiplication, where the cores contain matrices as elements instead of scalar values. Just like multiplying two matrices, we compute the tensor products of the corresponding elements and then sum over the columns and rows, respectively. We will use the core notation to derive compact representations of tensor trains and tensor-train operators. Furthermore, for given TT cores (3.4.2) and (3.4.3), we define the *rank-transposed* cores $\left[\mathbf{T}^{(i)}\right]^{\mathbb{T}} \in \mathbb{R}^{r_i \times n_i \times r_{i-1}}$ and $\left[\mathbf{G}^{(i)}\right]^{\mathbb{T}} \in \mathbb{R}^{R_i \times m_i \times n_i \times R_{i-1}}$ as

$$\left[\mathbf{T}^{(i)}\right]^{\mathbb{T}} = \begin{bmatrix} \mathbf{T}^{(i)}_{1,:,:,1} & \cdots & \mathbf{T}^{(i)}_{r_{i-1},:,:,1} \\ \vdots & \ddots & \vdots \\ \mathbf{T}^{(i)}_{1,:,:,r_i} & \cdots & \mathbf{T}^{(i)}_{r_{i-1},:,:,r_i} \end{bmatrix},$$

and

$$\left[\mathbf{A}^{(i)}\right]^{\mathbb{T}} = \begin{bmatrix} \mathbf{A}^{(i)}_{1,:,:,1} & \cdots & \mathbf{A}^{(i)}_{s_{i-1},:,:,1} \\ \vdots & \ddots & \vdots \\ \mathbf{A}^{(i)}_{1,:,:,s_i} & \cdots & \mathbf{A}^{(i)}_{s_{i-1},:,:,s_i} \end{bmatrix}. \tag{3.4.4}$$

Note that the vectors/matrices within the cores are not transposed, only the outer indices of each element are interchanged.

3.4.2. Addition and Multiplication

In this section, we will only focus on TT operators $\mathbf{G} \in \mathbb{R}^{M \times N}$ since the calculation rules can then be applied to any tensor $\mathbf{T} \in \mathbb{R}^N$ by considering $\overline{\mathbf{T}} \in \mathbb{R}^{N \times \mathbb{1}}$ with $\mathbb{1} = (1, \ldots, 1)^T$, see Section 2.1. For a TT decomposition $\mathbf{T}$ with cores $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$, the cores of $\overline{\mathbf{T}}$ are defined as $\overline{\mathbf{T}}^{(i)} \in \mathbb{R}^{r_{i-1} \times n_i \times 1 \times r_i}$ with

$$\overline{\mathbf{T}}^{(i)}_{k_{i-1},x_i,1,k_i} = \mathbf{T}_{k_{i-1},x_i,k_i},$$

where we just inserted a mode with mode size 1.

As it is the case for the ranks of canonical tensors, the TT ranks of $\mathbf{G} = \mathbf{G}_1 + \mathbf{G}_2$ are bounded by the sum of the TT ranks of $\mathbf{G}_1 \in \mathbb{R}^{M\times N}$ and $\mathbf{G}_2 \in \mathbb{R}^{M\times N}$.

Theorem 3.4.4. *For tensor operators $\mathbf{G}_1, \mathbf{G}_2 \in \mathbb{R}^{M\times N}$ with TT representations*

$$\mathbf{G}_1 = \left[\mathbf{G}_1^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{G}_1^{(d)}\right], \quad \mathbf{G}_2 = \left[\mathbf{G}_2^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{G}_2^{(d)}\right],$$

the sum $\mathbf{G} = \mathbf{G}_1 + \mathbf{G}_2$ is given by

$$\begin{aligned}\mathbf{G} = \left[\left[\mathbf{G}_1^{(1)}\right] \left[\mathbf{G}_2^{(1)}\right]\right] \otimes \begin{bmatrix}\left[\mathbf{G}_1^{(2)}\right] & 0 \\ 0 & \left[\mathbf{G}_2^{(2)}\right]\end{bmatrix} \otimes \cdots \\ \cdots \otimes \begin{bmatrix}\left[\mathbf{G}_1^{(d-1)}\right] & 0 \\ 0 & \left[\mathbf{G}_2^{(d-1)}\right]\end{bmatrix} \otimes \begin{bmatrix}\left[\mathbf{G}_1^{(d)}\right] \\ \left[\mathbf{G}_2^{(d)}\right]\end{bmatrix}.\end{aligned}$$

Proof. For $i \in \{2, \ldots, d-1\}$, it holds that

$$\begin{bmatrix}\left[\mathbf{G}_1^{(i)}\right] & 0 \\ 0 & \left[\mathbf{G}_2^{(i)}\right]\end{bmatrix} \otimes \begin{bmatrix}\left[\mathbf{G}_1^{(i+1)}\right] \otimes \cdots \otimes \left[\mathbf{G}_1^{(d)}\right] \\ \left[\mathbf{G}_2^{(i+1)}\right] \otimes \cdots \otimes \left[\mathbf{G}_2^{(d)}\right]\end{bmatrix} = \begin{bmatrix}\left[\mathbf{G}_1^{(i)}\right] \otimes \cdots \otimes \left[\mathbf{G}_1^{(d)}\right] \\ \left[\mathbf{G}_2^{(i)}\right] \otimes \cdots \otimes \left[\mathbf{G}_2^{(d)}\right]\end{bmatrix}.$$

Thus, we obtain

$$\begin{aligned}\mathbf{G} &= \left[\left[\mathbf{G}_1^{(1)}\right] \left[\mathbf{G}_2^{(1)}\right]\right] \otimes \begin{bmatrix}\left[\mathbf{G}_1^{(2)}\right] \otimes \cdots \otimes \left[\mathbf{G}_1^{(d)}\right] \\ \left[\mathbf{G}_2^{(2)}\right] \otimes \cdots \otimes \left[\mathbf{G}_2^{(d)}\right]\end{bmatrix} \\ &= \left[\mathbf{G}_1^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{G}_1^{(d)}\right] + \left[\mathbf{G}_2^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{G}_2^{(d)}\right]. \qquad \square\end{aligned}$$

The TT ranks of the product of two tensor-train operators $\mathbf{G} \in \mathbb{R}^{M\times N}$ and $\mathbf{H} \in \mathbb{R}^{N\times P}$ are bounded by the products of the TT ranks of $\mathbf{G}$ and $\mathbf{H}$, see Theorem 3.4.5. The multiplication of two tensor-train operators is depicted in Figure 3.4.

Theorem 3.4.5. *For two tensor operators $\mathbf{G}_1 \in \mathbb{R}^{M\times N}$ and $\mathbf{G}_2 \in \mathbb{R}^{N\times P}$ with TT representations*

$$\mathbf{G}_1 = \left[\mathbf{G}_1^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{G}_1^{(d)}\right], \quad \mathbf{G}_2 = \left[\mathbf{G}_2^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{G}_2^{(d)}\right],$$

the TT cores of the product $\mathbf{G} = \mathbf{G}_1 \cdot \mathbf{G}_2 = \left[\mathbf{G}^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{G}^{(d)}\right]$ *are given by*

$$\mathbf{G}^{(i)}_{\overline{k_{i-1},l_{i-1}},:,:,\overline{k_i,l_i}} = \left(\mathbf{G}_1^{(i)}\right)_{k_{i-1},:,:,k_i} \cdot \left(\mathbf{G}_2^{(i)}\right)_{l_{i-1},:,:,l_i},$$

for $i = 1, \ldots, d$.

Proof. Assume $\mathbf{G}_1$ has TT ranks $r_0, \ldots r_d$ and $\mathbf{G}_2$ has TT ranks $s_0, \ldots, s_d$. Then we obtain

$$\begin{aligned}
\mathbf{G} &= \sum_{k_0=1}^{r_0} \cdots \sum_{k_d=1}^{r_d} \sum_{l_0=1}^{s_0} \cdots \sum_{l_d=1}^{s_d} \mathbf{G}^{(1)}_{\overline{k_0,l_0},:,:,\overline{k_1,l_1}} \otimes \cdots \otimes \mathbf{G}^{(d)}_{\overline{k_{d-1},l_{d-1}},:,:,\overline{k_d,l_d}} \\
&= \sum_{k_0=1}^{r_0} \cdots \sum_{k_d=1}^{r_d} \left(\mathbf{G}_1^{(1)}\right)_{k_0,:,:,k_1} \otimes \cdots \otimes \left(\mathbf{G}_1^{(d)}\right)_{k_{d-1},:,:,k_d} \\
&\quad \cdot \sum_{l_0=1}^{s_0} \cdots \sum_{l_d=1}^{s_d} \left(\mathbf{G}_2^{(1)}\right)_{l_0,:,:,l_1} \otimes \cdots \otimes \left(\mathbf{G}_2^{(d)}\right)_{l_{d-1},:,:,l_d}.
\end{aligned}$$

□

The computation of the product $\mathbf{G} = \mathbf{G}_1 \cdot \mathbf{G}_2$ can be implemented efficiently using Algorithm 4 from [42].

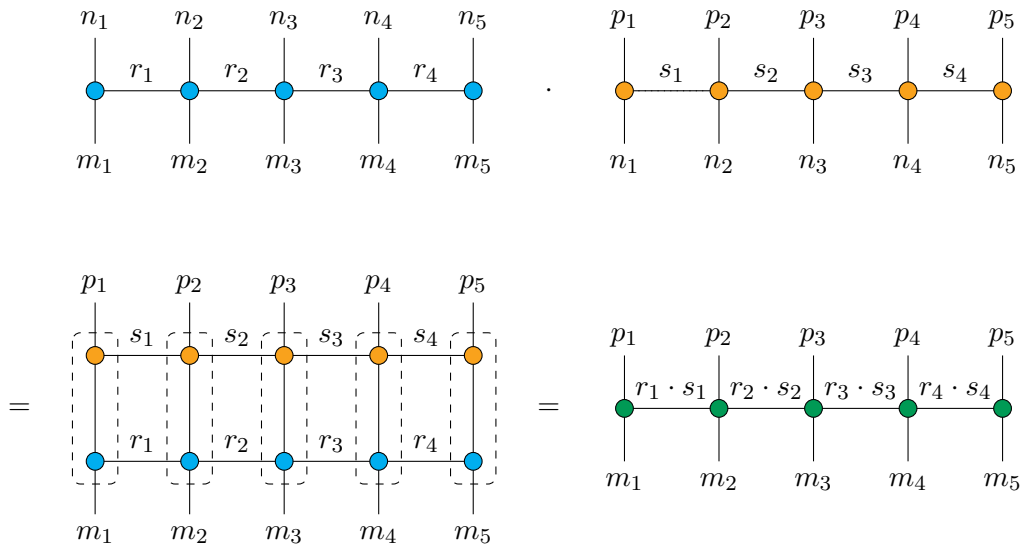


Figure 3.4: *Multiplication of two tensor-train operators: Two TT operators* $\mathbf{G} \in \mathbb{R}^{M \times N}$ *(blue) and* $\mathbf{H} \in \mathbb{R}^{N \times P}$ *(orange) of order* 10 *are multiplied by contracting the modes* $n_1, \ldots, n_d$*, which is depicted by joining corresponding arms. The result (green) is a tensor in* $\mathbb{R}^{M \times P}$.

3.4.3. Orthonormalization

In Section 2.6, we introduced the notion of orthonormality for tensors. Now, we extend the definition to TT decompositions $\mathbf{T} \in \mathbb{R}^N$. With respect to Definition 2.4.2, we consider a TT core $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$ and its matricizations

$$\mathcal{L}\left(\mathbf{T}^{(i)}\right) = \mathbf{T}^{(i)}\Big|_{r_{i-1},n_i}^{r_i} \qquad \text{and} \qquad \mathcal{R}\left(\mathbf{T}^{(i)}\right) = \mathbf{T}^{(i)}\Big|_{r_{i-1}}^{n_i,r_i}. \tag{3.4.5}$$

The matricization $\mathcal{L}\left(\mathbf{T}^{(i)}\right)$ is called the *left-unfolding* of $\mathbf{T}^{(i)}$ and $\mathcal{R}\left(\mathbf{T}^{(i)}\right)$ is called the *right-unfolding* of $\mathbf{T}^{(i)}$, cf. [37]. We will use these specific matricizations several times in the next chapters. A TT core is called *left-orthonormal* if its left-unfolding is orthonormal with respect to the columns, i.e.

$$\left(\mathcal{L}\left(\mathbf{T}^{(i)}\right)\right)^T \cdot \mathcal{L}\left(\mathbf{T}^{(i)}\right) = \mathbf{T}^{(i)}\Big|_{r_i}^{r_{i-1},n_i} \cdot \mathbf{T}^{(i)}\Big|_{r_{i-1},n_i}^{r_i} = I \in \mathbb{R}^{r_i \times r_i}.$$

A TT core whose right-unfolding is orthonormal with respect to the rows, i.e.

$$\mathcal{R}\left(\mathbf{T}^{(i)}\right) \cdot \left(\mathcal{R}\left(\mathbf{T}^{(i)}\right)\right)^T = \mathbf{T}^{(i)}\Big|_{r_{i-1}}^{n_i,r_i} \cdot \mathbf{T}^{(i)}\Big|_{n_i,r_i}^{r_{i-1}} = I \in \mathbb{R}^{r_{i-1} \times r_{i-1}},$$

is called *right-orthonormal*. We also call TT decompositions of tensors $\mathbf{T} \in \mathbb{R}^N$ left- and right-orthonormal, respectively. See Figure 3.5 for details.

Algorithms for the left- and right-orthonormalization, respectively, can be found in Appendix A.2. Left-orthonormalization, see Algorithm 9, produces a tensor train where all cores except for the last one are left-orthonormal whereas right-orthonormalization, see Algorithm 10, produces a tensor train where all cores except for the first one are right-orthonormal. Figure 3.6 shows a visualization of the left-orthonormalization of a TT decomposition. Note that a tensor train $\mathbf{T}$ remains the same if we apply the Algorithms 9 or 10 to it. The algorithms simply compute a different but equivalent representation. Additionally, left- and right-orthonormalization of a TT decomposition may decrease the TT ranks due to the applied SVDs.

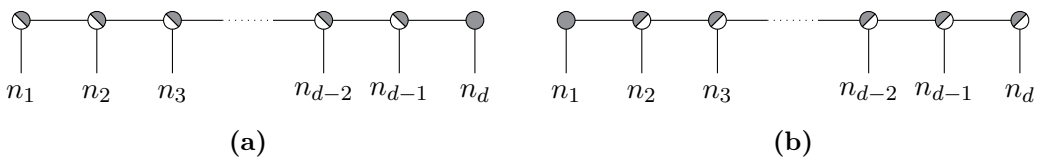


Figure 3.5: *Orthonormal tensor trains: (a) Left-orthonormal tensor train with first $d-1$ cores being left-orthonormal. (b) Right-orthonormal tensor train with last $d-1$ cores being right-orthonormal. See Section 2.6 for an explanation of the graphical notation.*

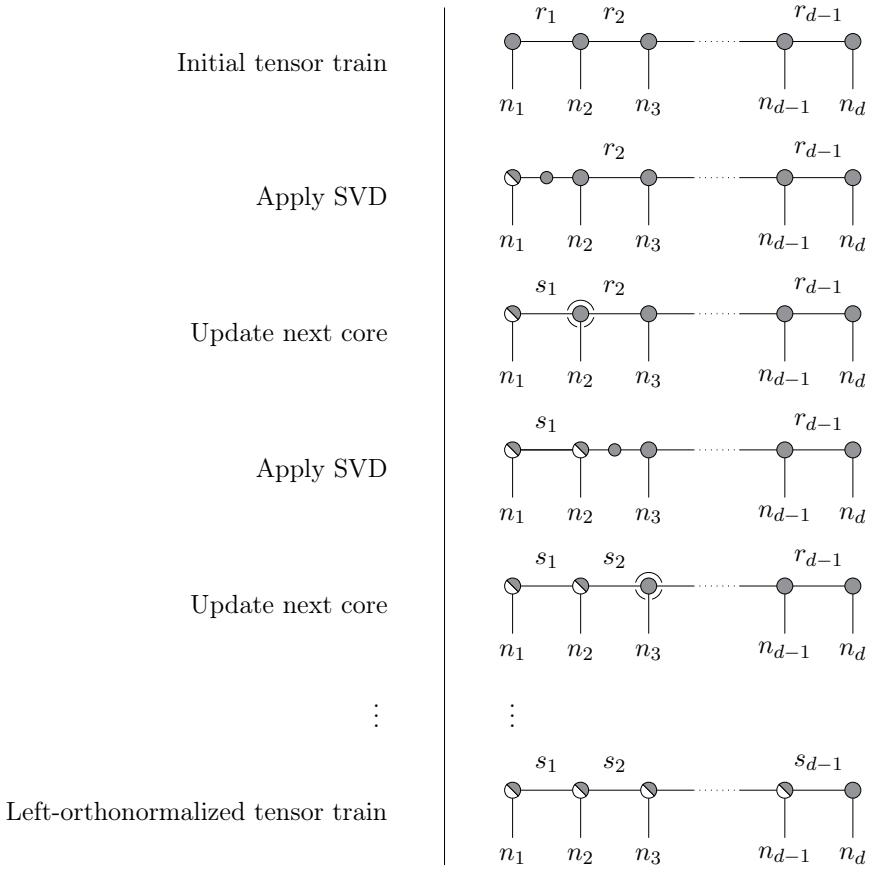


Figure 3.6: *Left-orthonormalization of a tensor train: After applying an SVD, the non-orthonormal part (depicted by a small circle) is shifted to the next core. This procedure is repeated until the first $d-1$ cores are left-orthonormal.*

3.4.4. Calculating Norms

Especially for estimating the errors of TT approximations in the following chapters, we will need to calculate norms of high-dimensional tensors in the TT format. In particular, we are interested in the 2-norm of a tensor train $\mathbf{T} \in \mathbb{R}^N$. As mentioned in Section 2.5, the p-norm of $\mathbf{T}$ is equal to the classical p-norm of its vectorization. However, since the storage consumption of $\mathrm{vec}(\mathbf{T})$ may be extremely high, we need to compute the norm of $\mathbf{T}$ in a more efficient way. From (2.5.3), we know it holds that $\|\mathbf{T}\|_2 = \sqrt{\mathbf{T}^T \cdot \mathbf{T}}$. Thus, one way to calculate $\|\mathbf{T}\|_2$ would be the computation of a sequence of tensor contractions, i.e. the TT cores of $\mathbf{T}$ and $\mathbf{T}^T$ are contracted stepwise from left to right. This is described in Algorithm 1.

In order to calculate the 2-norm of $\mathbf{T}$, we store the tensor train $\mathbf{T}$ itself and, additionally, compute a tensor C of order 4 and a matrix D in each step, see Algorithm 1. The storage of C and D can be estimated as $O(r^4)$, where r is the maximum of all TT ranks of $\mathbf{T}$ and n is the maximum of all mode sizes in N. Thus, the storage consumption of Algorithm 1 is estimated as $O(r \cdot n \cdot d + r^4)$.

Algorithm 1 Computation of the 2-norm of tensor trains

Input: Tensor train $\mathbf{T} \in \mathbb{R}^N$ with cores $\mathbf{T}^{(1)}, \dots, \mathbf{T}^{(d)}$ and ranks $r_0, \dots, r_d$.
Output: 2-norm of $\mathbf{T}$.

1: Define $D \in \mathbb{R}^{r_1 \times r_1'}$, $r_1 = r_1'$, by $D = \left\langle \mathbf{T}^{(1)}, \mathbf{T}^{(1)} \right\rangle_{r_0, n_1}$, where r_1 is the rank index of the first argument and r_1' is the rank index of the second argument.
2: **for** $i = 2, ..., d$ **do**
3: Define $C \in \mathbb{R}^{r_{i-1} \times r_i \times r_{i-1}' \times r_i'}$ by $C = \left\langle \mathbf{T}^{(i)}, \mathbf{T}^{(i)} \right\rangle_{n_i}$.
4: Set D to $\langle D, C \rangle_{r_{i-1}, r_{i-1}'}$, resulting in $D \in \mathbb{R}^{r_i \times r_i'}$.
5: **end for**
6: Set $\|\mathbf{T}\|_2 = \sqrt{D}$.

If the given tensor $\mathbf{T}$ is left- or right-orthonormal, respectively, we can apply an even more efficient technique to compute $\|\mathbf{T}\|_2$. If $\mathbf{T}$ is left-orthonormal, contracting the first $d-1$ cores results in an identity matrix in $\mathbb{R}^{r_{d-1} \times r_{d-1}}$. If $\mathbf{T}$ is right-orthonormal, contracting the last $d-1$ cores results in an identity matrix in $\mathbb{R}^{r_1 \times r_1}$. See Figure 3.7 for a visualization. That is, the 2-norm of $\mathbf{T}$ is then given by $\left\| \mathcal{L}\left(\mathbf{T}^{(d)}\right) \right\|_2$ and $\left\| \mathcal{R}\left(\mathbf{T}^{(1)}\right) \right\|_2$, respectively.

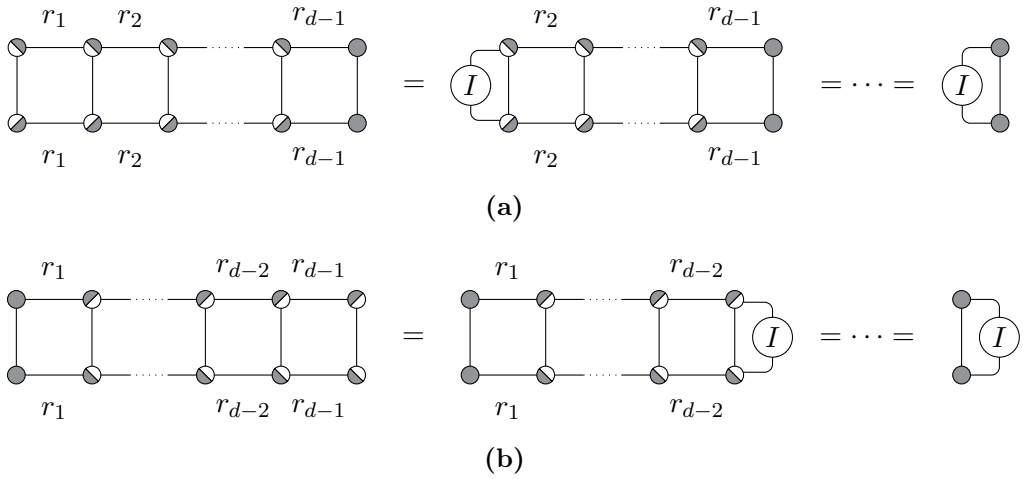


Figure 3.7: *Calculating the 2-norm of a tensor train: (a) For left-orthonormal tensor trains, the contraction of the first $d-1$ cores successively yields an identity matrix. (b) Correspondingly, for right-orthonormal tensor trains, the contraction of the last $d-1$ cores successively yields an identity matrix.*

Moreover, we are interested in the 1-norm of tensor trains $\mathbf{T} \in \mathbb{R}^N$. In particular, we will compute TT approximations of tensors representing probability distributions which should satisfy $\|\mathbf{T}\|_1 = 1$. Unfortunately, there is no algorithm available for an efficient computation of the 1-norm. However, if we assume that all entries of $\mathbf{T}$ are non-negative – or that the absolute values of all negative entries are small

enough to neglect them – we can calculate/approximate $\|\mathbf{T}\|_1$ simply by

$$\|\mathbf{T}\|_1 \approx \mathbf{T}^T \cdot ([\mathbb{1}_1] \otimes \cdots \otimes [\mathbb{1}_d]),$$

with $\mathbb{1}_i = (1, \ldots, 1)^T \in \mathbb{R}^{n_i}$.

3.4.5. Conversion

As stated by the Eckart–Young theorem [72], computing a truncated SVD of a matrix $A \in \mathbb{R}^{m \times n}$ with $m \geq n$, i.e. $\tilde{A} = U \Sigma V^T$ where Σ only contains the r largest singular values of A, yields the best rank-r approximation of A with respect to the Frobenius norm. Although there is no such clear statement for orders $d > 2$, we generalize that concept and apply a sequence of SVDs in order to convert a given tensor $\mathbf{T} \in \mathbb{R}^N$ from full format into the TT format. Algorithm 2, which was presented in [42], can be used to compute an exact ($\varepsilon = 0$) or an approximated ($\varepsilon > 0$) TT decomposition of $\mathbf{T}$, respectively. Due to the similar structure to Algorithm 9, the algorithm below produces a left-orthonormal tensor train. The diagrammatic notation of Algorithm 2 is shown in Figure 3.8.

Algorithm 2 Conversion of tensors in full format into the TT format

Input: Tensor $\mathbf{T} \in \mathbb{R}^N$ in full format and a threshold ε.

Output: Approximation $\mathbf{U}$ of $\mathbf{T}$ in TT format with cores $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(d)}$ and ranks $r_0, \ldots, r_d$.

1: Set $r_0 = 1$ and $r_d = 1$.
2: **for** $k = 1, ..., d-1$ **do**
3: $\quad A = \mathbf{T} \Big|^{n_{k+1},\ldots,n_d}_{r_{k-1},n_k}$.
4: $\quad$ Compute SVD of A, i.e. $A = U \Sigma V^T$ with $\Sigma \in \mathbb{R}^{s \times s}$.
5: $\quad$ Set $r_k \leq s$ to the smallest index such that $\sqrt{\sigma_{r_{k+1}}^2 + \cdots + \sigma_s^2} \leq \varepsilon$.
6: $\quad$ Discard rows and columns of U, Σ, and V corresponding to singular values $\sigma_{r_{k+1}}, \ldots, \sigma_s$.
7: $\quad$ Define $\mathbf{U}^{(k)} \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ such that $\mathcal{L}\left(\mathbf{U}^{(k)}\right) = U$.
8: $\quad$ Define remainder $\mathbf{T} = \Sigma V^T \in \mathbb{R}^{r_k \times n_{k+1} \cdots n_d}$.
9: **end for**
10: Set d-th core to $\mathbf{U}^{(d)}_{:,:,1} = \mathbf{T}$.

At each step the matrix A is approximated by a truncated SVD using only the singular values $\sigma_1, \ldots, \sigma_{r_k}$. Assume we calculate an SVD of A, i.e. $A = U \Sigma V^T$ with $\Sigma \in \mathbb{R}^{s \times s}$, and split Σ into $\Sigma_1 = \operatorname{diag}(\sigma_1, \ldots, \sigma_{r_k}, 0, \ldots, 0)$ and $\Sigma_2 = \operatorname{diag}(0, \ldots, 0, \sigma_{r_{k+1}}, \ldots, \sigma_s)$. Then, the Frobenius norm of $A - \tilde{A}$ with

$\tilde{A} = U\,\Sigma_1\,V^T$ is given by

$$\left\|A - \tilde{A}\right\|_F = \left\|U\,\Sigma_2\,V^T\right\|_F = \|\Sigma_2\|_F = \sqrt{\sigma_{r_{k+1}}^2 + \cdots + \sigma_s^2}.$$

It was shown in [42] that the TT approximation $\mathbf{U}$ – obtained by applying Algorithm 2 with threshold ε to a tensor $\mathbf{T}$ – satisfies $\|\mathbf{T} - \mathbf{U}\|_2 \leq \varepsilon\sqrt{d-1}$.

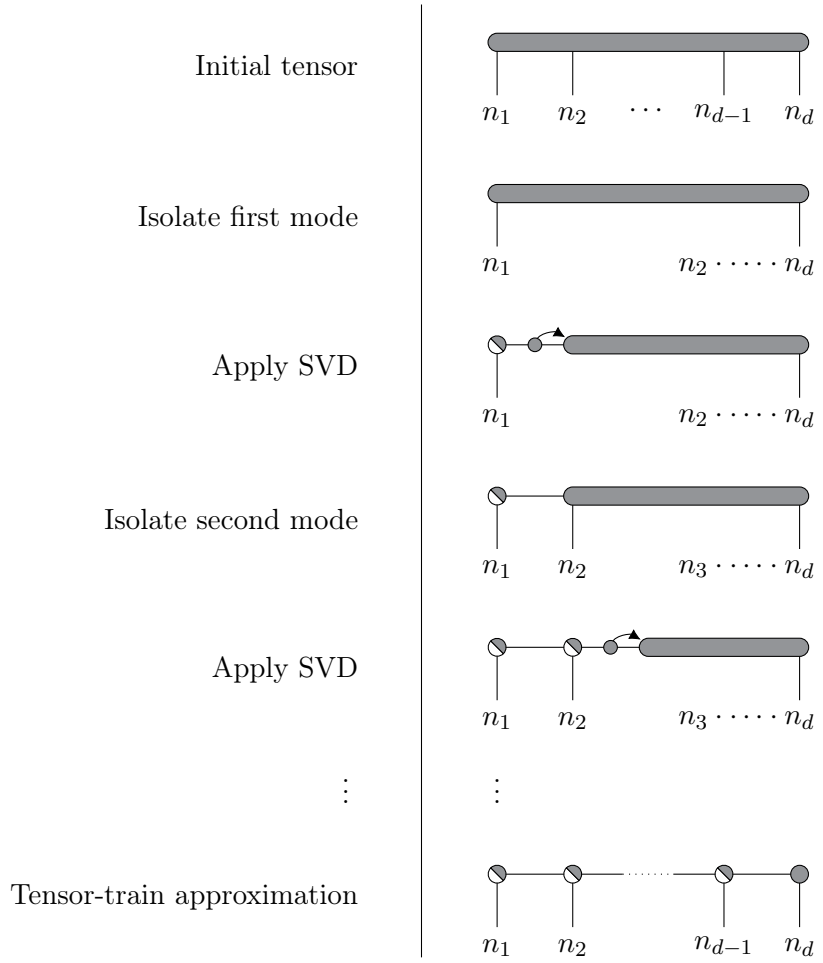


Figure 3.8: *Conversion from full format into TT format: By reshaping and applying SVDs, one mode is isolated in every step. The matrix U of an SVD defines the separated TT core while the matrices Σ (depicted by the small circles) and V (depicted by the bars) define the remainder.*

The principle shown in Algorithm 2 can also be used to convert a given operator $\mathbf{G} \in \mathbb{R}^{M \times N}$ into the TT format. In general, we do, however, not want to compute a tensor in full format and then convert it to the TT format – especially we do not convert full representations of linear operators. Instead, we construct tensor operators describing a specific system directly as a tensor train, i.e. we compute an exact TT decomposition. In this way, we do not have to deal with the storage consumption of tensor operators in full format. Additionally, all the numerical computations should ideally be directly carried out in the TT format such that we automatically compute low-rank approximations without necessitating the conversion to the TT format.

In what follows, we will particularly consider tensor operators whose canonical representations are known. The aim is then to convert these representations into the TT format obtaining low-rank decompositions. If a tensor operator $\mathbf{G}^{M \times N}$ can be written in canonical format as

$$\mathbf{G} = \sum_{k=1}^{r} \mathbf{G}^{(1)}_{k,:,:} \otimes \cdots \otimes \mathbf{G}^{(d)}_{k,:,:},$$

it can be represented in the TT format as

$$\mathbf{G} = \begin{bmatrix} \left(\mathbf{G}^{(1)}\right)_{1,:,:} \cdots \left(\mathbf{G}^{(1)}\right)_{r,:,:} \end{bmatrix} \otimes \begin{bmatrix} \left(\mathbf{G}^{(2)}\right)_{1,:,:} & & 0 \\ & \ddots & \\ 0 & & \left(\mathbf{G}^{(2)}\right)_{r,:,:} \end{bmatrix} \otimes \cdots$$
$$\cdots \otimes \begin{bmatrix} \left(\mathbf{G}^{(d-1)}\right)_{1,:,:} & & 0 \\ & \ddots & \\ 0 & & \left(\mathbf{G}^{(d-1)}\right)_{r,:,:} \end{bmatrix} \otimes \begin{bmatrix} \left(\mathbf{G}^{(d)}\right)_{1,:,:} \\ \vdots \\ \left(\mathbf{G}^{(d)}\right)_{r,:,:} \end{bmatrix}.$$

Except for the first and the last rank, the ranks of this decomposition are all equal to r. However, r is just an upper bound for the TT ranks. In specific cases, there exist low-rank TT decompositions of $\mathbf{G}$ with ranks much smaller than r. We will consider such TT operators particularly in Section 6.

Example 3.4.6. *As a simple example for a tensor whose TT ranks are smaller than its canonical rank, consider the matrices*

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad J = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

and the canonical tensor

$$\mathbf{G} \;=\; J \otimes I \otimes I \;+\; I \otimes J \otimes I \;+\; I \otimes I \otimes J. \tag{3.4.6}$$

The canonical rank of $\mathbf{G}$ *is* $r = 3$*. However,* $\mathbf{G}$ *can be represented as a tensor-train operator with decomposition*

$$\mathbf{G} = \begin{bmatrix} J & I \end{bmatrix} \otimes \begin{bmatrix} I & 0 \\ J & I \end{bmatrix} \otimes \begin{bmatrix} I \\ J \end{bmatrix}, \tag{3.4.7}$$

which has TT ranks $1, 2, 2, 1$.

The TT representation given in (3.4.7) is a special case of the more general SLIM decomposition, which we will present in Section 6. If we consider higher orders for $\mathbf{G}$ and – at the same time – keep the alternating canonical structure (3.4.6), the ranks of the TT representation (obtained by increasing the number of middle cores in (3.4.7)) do not grow, i.e. the TT ranks are then given by $1, 2, 2, \ldots, 2, 1$. This property will be investigated in detail later. Moreover, the Example 3.4.6 above shows that conversions from the TT format into the canonical format may lead to very large canonical ranks.

For the sake of completeness, we also present a method to convert a given Tucker tensor into the TT format. Given a tensor $\mathbf{T} \in \mathbb{R}^N$ in the Tucker format with Tucker factors $\mathbf{T}_{\mathrm{TF}}^{(i)} \in \mathbb{R}^{n_i \times r_i}$, $i = 1, \ldots, d$, and Tucker core $\mathbf{U} \in \mathbb{R}^{r_1 \times \cdots \times r_d}$, a TT decomposition of $\mathbf{T}$ can be obtained by computing a TT representation of the Tucker core. Consider

$$\begin{aligned} \mathbf{T} &= \left(\mathbf{T}_{\mathrm{TF}}^{(1)} \otimes \cdots \otimes \mathbf{T}_{\mathrm{TF}}^{(d)} \right) \cdot \mathbf{U} \\ &= \left(\mathbf{T}_{\mathrm{TF}}^{(1)} \otimes \cdots \otimes \mathbf{T}_{\mathrm{TF}}^{(d)} \right) \cdot \left(\left[\mathbf{U}^{(1)} \right] \otimes \cdots \otimes \left[\mathbf{U}^{(d)} \right] \right), \end{aligned}$$

with $\mathbf{U}^{(i)} \in \mathbb{R}^{s_{i-1} \times r_i \times s_i}$. The decomposition $\left[\mathbf{U}^{(1)} \right] \otimes \cdots \otimes \left[\mathbf{U}^{(d)} \right]$ is a TT representation of the Tucker core $\mathbf{U}$. Following the multiplication rules, the contraction of the modes of the cores of $\mathbf{T}_{\mathrm{TF}}$ and $\mathbf{U}$ results in

$$\mathbf{T}_{k_{i-1},:,k_i}^{(i)} = \mathbf{T}_{\mathrm{TF}}^{(i)} \cdot \mathbf{U}_{k_{i-1},:,k_i}^{(i)},$$

for $i = 1, \ldots, d$ and $k_i = 1, \ldots, s_i$. The ranks of the TT decomposition of $\mathbf{T}$ are equal to the TT ranks of $\mathbf{U}$, i.e. if we find a low-rank representation for the Tucker core, we can provide a low-rank representation for the whole tensor in TT format.

3.5. Modified Tensor-Train Formats

In this section, we will consider three different tensor representations which are based on slight modifications of the TT format, namely the *quantized tensor-train format*, the *block tensor-train format*, and the *cyclic tensor-train format*. In short, we will call these representations QTT, BTT, and CTT format, respectively.

3.5.1. Quantized Tensor-Train Format

The QTT format [73] can be approached from different perspectives. On the one hand, it can be regarded as the TT decomposition of an appropriately quantized tensor, i.e. given a tensor $\mathbf{G} \in \mathbb{R}^{M \times N}$ and factorizations of the modes

$$m_i = m_{i,1} \cdot \ldots \cdot m_{i,c_i} \quad \text{and} \quad n_i = n_{i,1} \cdot \ldots \cdot n_{i,c_i},$$

with $m_{i,j}, n_{i,j} \in \mathbb{N}$ for $i = 1, \ldots, d$ and $j = 1, \ldots, c_i$, we define the corresponding quantization

$$\mathbf{G}' \in \mathbb{R}^{(m_{1,1} \times n_{1,1}) \times \cdots \times (m_{1,c_1} \times n_{1,c_1}) \times \cdots \times (m_{d,1} \times n_{d,1}) \times \cdots \times (m_{d,c_d} \times n_{1,c_d})}$$

by

$$\mathbf{G}'_{x_{1,1}, y_{1,1}, \ldots, x_{1,c_1}, y_{1,c_1}, \ldots, x_{d,1}, y_{d,1}, \ldots, x_{d,c_d}, y_{d,c_d}} = \mathbf{G}_{x_1, y_1, \ldots, x_d, y_d},$$

where $x_i = \overline{x_{i,1}, \ldots, x_{i,c_i}}$ and $y_i = \overline{y_{i,1}, \ldots, y_{i,c_i}}$ for $i = 1, \ldots, d$. A TT decomposition of $\mathbf{G}'$ then represents a QTT decomposition of the former tensor $\mathbf{G}$. The ranks of this decomposition are called QTT ranks. On the other hand, we can construct a QTT decomposition of $\mathbf{G}$ directly from its TT decomposition. Here, the TT cores $\mathbf{G}^{(i)}$, $i = 1, \ldots, d$, are divided into c_i smaller cores, see Algorithm 3 and Figure 3.9.

Algorithm 3 Conversion of TT operators into the QTT format

Input: TT operator $\mathbf{G} \in \mathbb{R}^{M \times N}$ with cores $\mathbf{G}^{(1)}, \ldots, \mathbf{G}^{(d)}$ and mode factorizations $m_i = m_{i,1} \cdot \ldots \cdot m_{i,c_i}$, $n_i = n_{i,1} \cdot \ldots \cdot n_{i,c_i}$ for $i = 1, \ldots, d$.

Output: QTT representation $\mathbf{G}_{\mathrm{QTT}}$ with cores $\mathbf{G}_{\mathrm{QTT}}^{i,j}$, $i = 1, \ldots, d$, $j = 1, \ldots, c_i$.

1: **for** $i = 1, \ldots, d$ **do**
2: Reshape $\mathbf{G}^{(i)} \in \mathbb{R}^{r_{i-1} \times m_i \times n_i \times r_i}$ to $\mathbf{H} \in \mathbb{R}^{r_{i-1} \times m_{i,1} \times n_{i,1} \times \cdots \times m_{i,c_i} \times n_{i,c_i} \times r_i}$.
3: Set $r_{i,0} = r_{i-1}$.
4: **for** $j = 1, \ldots, c_i - 1$ **do**
5: Set $H = \mathbf{H} \Big|^{m_{i,j+1}, n_{i,j+1}, \ldots, m_{i,c_i}, n_{i,c_i}, r_i}_{r_{i,j-1}, m_{i,j}, n_{i,j}}$.
6: Compute SVD of H, i.e. $M = U \Sigma V^T$ with $\Sigma \in \mathbb{R}^{r_{i,j} \times r_{i,j}}$.
7: Set the QTT core $\mathbf{G}_{\mathrm{QTT}}^{(i,j)}$ to a reshaped version of U with $\left(\mathbf{G}_{\mathrm{QTT}}^{(i,j)}\right)_{k_{i,j-1}, x_{i,j}, y_{i,j}, k_{i,j}} = U_{\overline{k_{i,j-1}, x_{i,j}, y_{i,j}}, k_{i,j}}$.
8: Define $\mathbf{H} = \Sigma V^T$ and reshape to $\mathbf{H} \in \mathbb{R}^{r_{i,j} \times m_{i,j+1} \times n_{i,j+1} \times \cdots \times m_{i,c_i} \times n_{i,c_i} \times r_i}$.
9: **end for**
10: Set $\mathbf{G}_{\mathrm{QTT}}^{(i,c_i)} = \mathbf{H}$
11: **end for**

Using the QTT format is only advantageous if the QTT ranks stay small. As it can be seen in Algorithm 3, this has not to be the case in general. The QTT ranks $r_{i,j}$ are only bounded by

$$\max\{r_{i,j-1} \cdot m_{i,j} \cdot n_{i,j}, \quad m_{i,j+1} \cdot n_{i,j+1} \cdot \dots \cdot m_{i,c_i} \cdot n_{i,c_i} \cdot r_i\},$$

and, thus, may increase rapidly during the quantization indicating that the TT cores $\mathbf{G}^{(i)}$ cannot be represented by low-rank QTT decompositions. However, we will consider a chemical system in Section 9.3.1 where the benefit of the QTT format in terms of numerical efficiency can be shown. Working with the QTT format, all properties from the TT format are directly applicable.

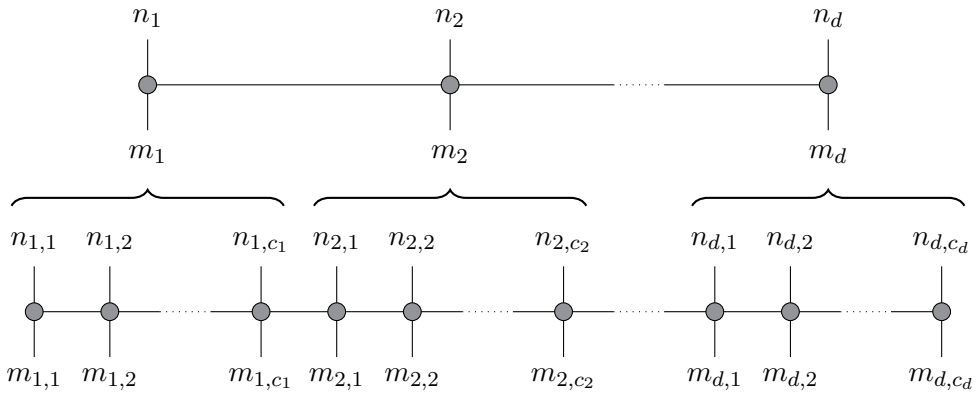


Figure 3.9: *Conversion from TT into QTT format: Each core is divided into several cores with smaller mode sizes.*

3.5.2. Block Tensor-Train Format

With the aid of the BTT format [19], several tensors can be represented simultaneously by one decomposition. Compared to the standard TT format, the idea here is to add an extra index to one of the TT cores. Hence, the tensor trains $\mathbf{T}_1, \dots, \mathbf{T}_b \in \mathbb{R}^N$ are said to be in the BTT format if they share $d-1$ cores and only differ in one core, i.e. for $k = 1, \dots, b$ we can write

$$\mathbf{T}_k = \left[\mathbf{T}^{(1)}\right] \otimes \dots \otimes \left[\mathbf{T}^{(p-1)}\right] \otimes \left[\mathbf{T}_k^{(p)}\right] \otimes \left[\mathbf{T}^{(p+1)}\right] \otimes \dots \otimes \left[\mathbf{T}^{(d)}\right], \tag{3.5.1}$$

where $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$ for $i \neq p$. The core $\mathbf{T}^{(p)}$ can be seen as a tensor of order 4 in $\mathbb{R}^{r_{p-1} \times n_p \times r_p \times b}$ with additional mode b. We then set $\mathbf{T}_k^{(p)} = \mathbf{T}_{:,:,:,k}^{(p)}$ and obtain the representation (3.5.1). Figure 3.10 shows a visualization of a tensor in BTT format.

By a combination of index permutations and decomposing/contracting the cores, the mode b can be shifted to a neighboring core of $\mathbf{T}^{(p)}$. Thus, we will write in short $\mathbf{T} \in \mathbb{R}^{N \times b}$ for a BTT decomposition $\mathbf{T}$ that represents b tensor trains $\mathbf{T}_1, \dots, \mathbf{T}_b$ at once. It will be clear from the context to which core the extra index b is attached.

We will use the BTT format in order to simultaneously approximate several eigentensors corresponding to an eigenvalue problem formulated in the TT format. This method will be described in detail in Section 4.3 and numerical examples will be given in Chapter 12.

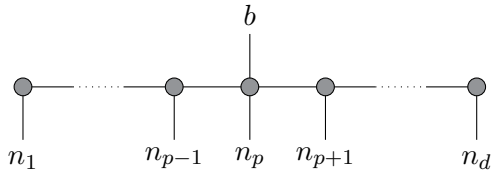


Figure 3.10: *Block tensor-train format: One of the cores has an extra index b such that the decomposition above represents b different tensor trains.*

3.5.3. Cyclic Tensor-Train Format

The last modified TT format we introduce in this work is the so-called CTT format or *cyclic matrix product states* [66]. Instead of requiring $r_0 = r_d = 1$ for the first and the last TT ranks, we now connect the cores in a cycle with $r_0 = r_d$. A CTT representation of a tensor $\mathbf{T}$ is then given by

$$\mathbf{T} = \sum_{k_1=1}^{r_1} \cdots \sum_{k_d=1}^{r_d} \mathbf{T}^{(1)}_{k_d,:,k_1} \otimes \cdots \otimes \mathbf{T}^{(d)}_{k_{d-1},:,k_d}, \tag{3.5.2}$$

with $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$ and CTT ranks $r_1, \ldots, r_d$. A graphical representation of the CTT format is shown in Figure 3.11.

The contraction of the rank indices $r_1, \ldots, r_{d-1}$ of all cores in (3.5.2) yields a super-core of the form

$$\begin{bmatrix} \mathbf{T}^{(1,\ldots,d)}_{1,:,\ldots,:,1} & \cdots & \mathbf{T}^{(1,\ldots,d)}_{1,:,\ldots,:,r_d} \\ \vdots & \ddots & \vdots \\ \mathbf{T}^{(1,\ldots,d)}_{r_d,:,\ldots,:,1} & \cdots & \mathbf{T}^{(1,\ldots,d)}_{r_d,:,\ldots,:,r_d} \end{bmatrix}, \tag{3.5.3}$$

where

$$\mathbf{T}^{(1,\ldots,d)}_{p,:,\ldots,:,q} = \sum_{k_1=1}^{r_1} \cdots \sum_{k_{d-1}=1}^{r_{d-1}} \mathbf{T}^{(1)}_{p,:,k_1} \otimes \cdots \otimes \mathbf{T}^{(d)}_{k_{d-1},:,q} \in \mathbb{R}^{n_1 \times \cdots \times n_d},$$

for $1 \leq p, q \leq r_d$. In order to obtain the full representation of $\mathbf{T}$, we then compute

the *trace* of the core given in (3.5.3), i.e.

$$\begin{aligned}\mathbf{T} &= \operatorname{tr}\left(\left[\mathbf{T}^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{T}^{(d)}\right]\right) \\ &= \sum_{k_d=1}^{r_d} \mathbf{T}^{(1,\ldots,d)}_{k_d,:,\ldots,:,k_d}.\end{aligned}$$

Due to the cyclic structure of a CTT representation of a tensor $\mathbf{T}$, we can easily change the ordering of the modes $n_1, \ldots, n_d$. Given a CTT decomposition as defined in (3.5.2), a cyclic permutation of the cores yields a tensor whose indices are permuted correspondingly. That is, if we define

$$\tilde{\mathbf{T}} = \operatorname{tr}\left(\left[\mathbf{T}^{(m)}\right] \otimes \cdots \otimes \left[\mathbf{T}^{(d)}\right] \otimes \left[\mathbf{T}^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{T}^{(m-1)}\right]\right), \tag{3.5.4}$$

with $1 \leq m \leq d$, we obtain

$$\tilde{\mathbf{T}}_{x_m,\ldots,x_d,x_1,\ldots,x_{m-1}} = \mathbf{T}_{x_1,\ldots,x_{m-1},x_m,\ldots,x_d}. \tag{3.5.5}$$

Of course, the consideration above also holds for the TT format and the canonical format since both can be seen as special cases of the QTT format. We will use the CTT format to express pseudoinverses in Section 7.

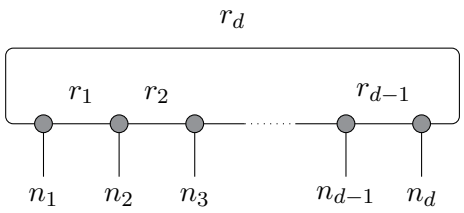


Figure 3.11: *Cyclic tensor-train format: Cores are coupled in a cycle, which can be seen as a generalization of the TT format by requiring that $r_0 = r_d \geq 1$.*

4
Optimization Problems in the Tensor-Train Format

In this chapter, we will explain how to solve optimization tasks in the TT format. There are two types of optimization problems of interest for us: systems of linear equations and eigenvalue problems. First, we will give an overview of different optimization algorithms proposed so far. After that, we will focus on two specific methods, namely the alternating linear scheme and the modified alternating linear scheme, see [37]. Based on the problem statement, we will consider algorithmic aspects as well as intrinsic properties, e.g. convergence criteria and computational complexity. Both schemes will provide the basis for the examination of high-dimensional systems in Part III of this thesis.

4.1. Overview

In order to solve optimization problems in the TT format such as systems of linear equations and eigenvalue problems, respectively, different algorithms have been proposed in recent years. Based on alternating optimizations of the TT cores of a given tensor train, the *alternating linear scheme* (ALS) [37] can be seen as the foundation of various methods. It optimizes a tensor train by constructing low-dimensional systems of linear equations (or eigenvalue problems) for each core, which then can be solved by standard numerical methods. Given an initial guess of the solution, ALS updates the TT cores successively during two bidirectional half sweeps. An intrinsic property of the ALS algorithm is that the TT ranks are fixed during the whole iteration, which can be either seen as an advantage or as a disadvantage. On the one hand, if our initial guess already has high TT ranks and we want to prevent that these ranks increase, ALS has lower computational costs than MALS and the accuracies of the computed approximations are comparable. On the other hand, if the TT ranks are low enough and can be adapted during the optimization steps, ALS may not be the first choice. In this case, one can employ algorithms such as, for instance, the *modified alternating linear scheme* (MALS) [37]. The basic principle of MALS is the optimization of two TT cores at once by calculating an optimized super-core, which represents the contraction of two adjacent TT cores. This super-core is then decomposed and the single components build the updated TT cores. This method is strongly related to the *density matrix renormalization group algorithm* from quantum mechanics [1].

Another way to adapt the TT ranks is the application of *alternating minimal energy methods* [74]. The idea of these algorithms is to expand the TT cores of an

initial guess by a residual tensor train, which is optimized by an ALS-like algorithm. However, the convergence observed in numerical experiments is comparable to the one of MALS, see [74]. Thus, in this work, we will focus on ALS and MALS.

4.2. (M)ALS for Systems of Linear Equations

This section is about the approximate solution of a system of linear equations given in TT format, where we assume that the corresponding TT operator is symmetric positive definite. A direct way to solve systems of linear equations represented by tensor trains would be to compute the corresponding matricizations and vectorizations, respectively, and express the linear equations in a classical way using matrices and vectors. Applying Algorithm 2 to the obtained solution would then yield an (approximate) solution in the TT format of the original system. However, our assumption is that the matricizations/vectorizations cannot be computed as the number of dimensions and/or the mode sizes are too large making it impossible to store those tensor unfoldings. In particular, the matricization of the TT operator can quickly consume a large amount of memory – even for a rather small number of dimensions. Thus, we need to compute an approximation of the solution of a given system directly in the TT format.

4.2.1. Problem Statement

We consider a system of linear equations

$$\mathbf{A} \cdot \mathbf{T} = \mathbf{U}, \tag{4.2.1}$$

which is given in the TT format with tensor trains $\mathbf{A} \in \mathbb{R}^{N \times N}$ and $\mathbf{T}, \mathbf{U} \in \mathbb{R}^{N}$. As before, the index set N is given by $N = (n_1, \dots, n_d)^T \in \mathbb{N}^d$ with $d \in \mathbb{N}$. The spaces $\mathbb{R}^N$ and $\mathbb{R}^{N \times N}$ are defined as in (2.1.2) and (2.1.3), respectively. We assume that the TT operator $\mathbf{A}$ is symmetric positive definite. That is, similar to classical linear algebra, it holds that

$$\begin{aligned} &\text{(i)} \quad \mathbf{A}_{x_1,y_1,x_2,y_2,\dots,x_d,y_d} = \mathbf{A}_{y_1,x_1,y_2,x_2,\dots,y_d,x_d} \text{ for any} \\ &\qquad X = (x_1, \dots, x_d) \in \mathcal{S} \text{ and } Y = (y_1, \dots, y_d) \in \mathcal{S}, \\ &\text{(ii)} \quad \mathbf{T}^T \cdot \mathbf{A} \cdot \mathbf{T} > 0 \text{ for any tensor } \mathbf{T} \in \mathbb{R}^N, \, \mathbf{T} \neq \mathbf{0}, \end{aligned} \tag{4.2.2}$$

where the space $\mathcal{S}$ is defined by

$$\mathcal{S} = \{1, \dots, n_1\} \times \{1, \dots, n_2\} \times \cdots \times \{1, \dots, n_d\},$$

and $\mathbf{0}$ denotes the tensor in $\mathbb{R}^N$ with $\mathbf{0}_{x_1,\dots,x_d} = 0$ for all $X = (x_1, \dots, x_d) \in \mathcal{S}$. The properties given in (4.2.2) are equivalent to

$$\begin{aligned} &\text{(i)} \quad A_{x,y} = A_{y,x} \text{ for } 1 \leq x, y \leq n_1 \cdot \ldots \cdot n_d, \\ &\text{(ii)} \quad v^T \cdot A \cdot v > 0 \text{ for any vector } v \in \mathbb{R}^{n_1 \cdot \ldots \cdot n_d},\ v \neq (0, \dots, 0)^T, \end{aligned}$$

where we consider the natural matricization A of the operator $\mathbf{A}$, see Section 2.4.

Theorem 4.2.1. *The solution of the system of linear equations* (4.2.1) *is the minimizer of the functional J given by*

$$J(\mathbf{T}) = \frac{1}{2}\, \mathbf{T}^T\, \mathbf{A}\, \mathbf{T} \;-\; \mathbf{U}^T\, \mathbf{T}. \tag{4.2.3}$$

Proof. From Theorem 2.4.5, we know that the equation (4.2.3) is equivalent to

$$\begin{aligned} J(\mathbf{T}) &= \frac{1}{2}\, \mathrm{vec}(\mathbf{T})^T \cdot \mathrm{mat}(\mathbf{A}) \cdot \mathrm{vec}(\mathbf{T}) - \mathrm{vec}(\mathbf{U})^T \cdot \mathrm{vec}(\mathbf{T}) \\ &= \frac{1}{2}\, \langle \mathrm{mat}(\mathbf{A}) \cdot \mathrm{vec}(\mathbf{T}), \mathrm{vec}(\mathbf{T}) \rangle - \langle \mathrm{vec}(\mathbf{U}), \mathrm{vec}(\mathbf{T}) \rangle, \end{aligned}$$

with $\langle \cdot, \cdot \rangle$ denoting the scalar product in Euclidean space. Thus, we can consider a classical system of the form $Av = w$ and the functional $J(v) = \frac{1}{2}\langle A \cdot v, v \rangle - \langle w, v \rangle$ in order to prove the assertion. Assume that the vector v is the solution of $Av = w$. Since A is symmetric positive definite, A is also invertible and, therefore, v is the unique solution. The gradient of J at point v is given by

$$\nabla J(v) = Av - w = 0.$$

The Hessian matrix of J is equal to the positive definite matrix A, which implies that v is the minimizer of the functional J. If, on the other hand, the vector v minimizes the functional J, it holds that $\nabla J(v) = 0$ and therefore $Av = w$. □

4.2.2. Retraction Operators

The key quantities which we will consider in order to describe ALS and MALS are the so-called retraction operators, see [37], which are defined in terms of a tensor train $\mathbf{T} \in \mathbb{R}^N$ with modes $n_1, \dots, n_d$ and TT ranks $r_0, \dots, r_d$.

Definition 4.2.2. *The retraction operator $\mathbf{Q}_i \in \mathbb{R}^{n_1 \times \cdots \times n_d \times m}$ for ALS with $m = r_{i-1} \cdot n_i \cdot r_i$, is defined as the tensor which results from replacing the i-th TT core of a given tensor train $\mathbf{T}$ with $\mathbf{I}^{(i)} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i \times m}$. The tensor $\mathbf{I}^{(i)}$ is*

given by

$$\mathbf{I}^{(i)}_{k_{i-1},x_i,k_i,\overline{l_{i-1},y_i,l_i}} = \delta_{k_{i-1},l_{i-1}} \cdot \delta_{x_i,y_i} \cdot \delta_{k_i,l_i}. \tag{4.2.4}$$

In (4.2.4), we use the notation introduced in Section 2.4. The tensor $\mathbf{I}^{(i)}$ can be seen as a permuted and reshaped version of an identity tensor, see Example 2.1.1. In order to describe the construction of the retraction operator $\mathbf{Q}_i$, we use the graphical representation introduced in Section 2.3, see Figure 4.1.

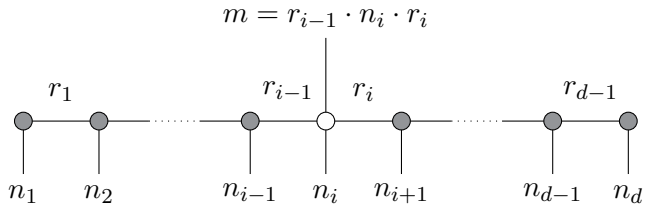


Figure 4.1: *Construction of the retraction operators for ALS: For the construction of* $\mathbf{Q}_i$*, the TT cores* $\mathbf{T}^{(j)}$ *(depicted by gray circles),* $j \neq i$*, are coupled with the tensor* $\mathbf{I}^{(i)}$ *of order 4 (depicted by a white circle).*

Contracting $\mathbf{Q}_i$ with a vector $v \in \mathbb{R}^m$ results in a tensor train $\mathbf{T}' \in \mathbb{R}^N$ which has the same cores as the tensor train $\mathbf{T}$ except for the i-th core. This TT core is replaced by a tensorized version of v, also called *folding* [37]. For the sake of simplicity, we write $\mathbf{Q}_i \cdot v$ instead of the contraction notation $\langle \mathbf{Q}_i, v \rangle_m$, see Section 2.2.2. We then obtain

$$\mathbf{Q}_i \cdot v = \left[\mathbf{T}^{(i)}\right] \otimes \cdots \otimes \left[\mathbf{T}^{(i-1)}\right] \otimes \left[\langle \mathbf{I}^{(i)}, v \rangle_m\right] \otimes \left[\mathbf{T}^{(i+1)}\right] \otimes \cdots \otimes \left[\mathbf{T}^{(d)}\right]. \tag{4.2.5}$$

with

$$\begin{aligned}
\left(\langle \mathbf{I}^{(i)}, v \rangle_m\right)_{k_{i-1},x_i,k_i} &= \sum_{l_{i-1}=1}^{r_{i-1}} \sum_{y_i=1}^{n_i} \sum_{l_i=1}^{r_i} \mathbf{I}^{(i)}_{k_{i-1},x_i,k_i,\overline{l_{i-1},y_i,l_i}} \cdot v_{\overline{l_{i-1},y_i,l_i}} \\
&= \sum_{l_{i-1}=1}^{r_{i-1}} \sum_{y_i=1}^{n_i} \sum_{l_i=1}^{r_i} \delta_{k_{i-1},l_{i-1}} \cdot \delta_{x_i,y_i} \cdot \delta_{k_i,l_i} \cdot v_{\overline{l_{i-1},y_i,l_i}} \\
&= v_{\overline{k_{i-1},x_i,k_i}}.
\end{aligned}$$

In a similar way, we define the retraction operator $\mathbf{Q}_{i,i+1}$, $i = 1, \ldots, d-1$, for MALS.

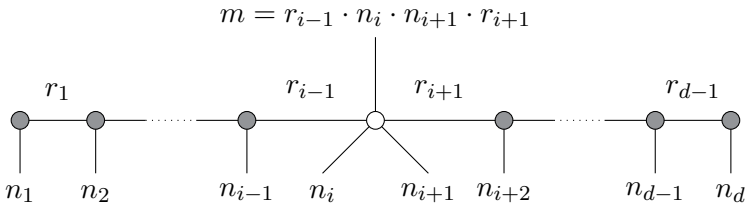


Figure 4.2: *Construction of the retraction operators for MALS: For the construction of* $\mathbf{Q}_{i,i+1}$*, the TT cores* $\mathbf{T}^{(j)}$ *(depicted by gray circles),* $j \neq i$ *and* $j \neq i+1$*, are coupled with the tensor* $\mathbf{I}^{(i,i+1)}$ *of order 5 (depicted by a white circle).*

Definition 4.2.3. *The retraction operator* $\mathbf{Q}_{i,i+1} \in \mathbb{R}^{n_1 \times \cdots \times n_d \times m}$ *for MALS,* $m = r_{i-1} \cdot n_i \cdot n_{i+1} \cdot r_{i+1}$*, is defined as the tensor which results from replacing the i-th and the* $(i+1)$*-th TT core of a given tensor train* $\mathbf{T}$ *with the tensor* $\mathbf{I}^{(i,i+1)} \in \mathbb{R}^{r_{i-1} \times n_i \times n_{i+1} \times r_{i+1} \times m}$*.* $\mathbf{I}^{(i,i+1)}$ *is given by*

$$\mathbf{I}^{(i,i+1)}_{k_{i-1},x_i,x_{i+1},k_{i+1},\overline{l_{i-1},y_i,y_{i+1},l_{i+1}}} = \delta_{k_{i-1},l_{i-1}} \cdot \delta_{x_i,y_i} \cdot \delta_{x_{i+1},y_{i+1}} \cdot \delta_{k_{i+1},l_{i+1}}. \tag{4.2.6}$$

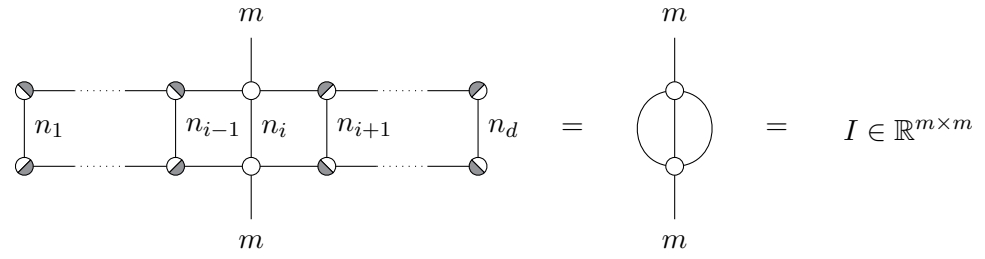
See Figure 4.2 for a visualization of $\mathbf{Q}_{i,i+1}$. Analogously to (4.2.5), we then define the product $\mathbf{Q}_{i,i+1} \cdot v$ with $v \in \mathbb{R}^m$ as

$$\mathbf{Q}_{i,i+1} \cdot v = \left[\mathbf{T}^{(i)}\right] \otimes \cdots \otimes \left[\mathbf{T}^{(i-1)}\right] \otimes \left[\mathbf{V}\right] \otimes \left[\mathbf{T}^{(i+1)}\right] \otimes \cdots \otimes \left[\mathbf{T}^{(d)}\right], \tag{4.2.7}$$

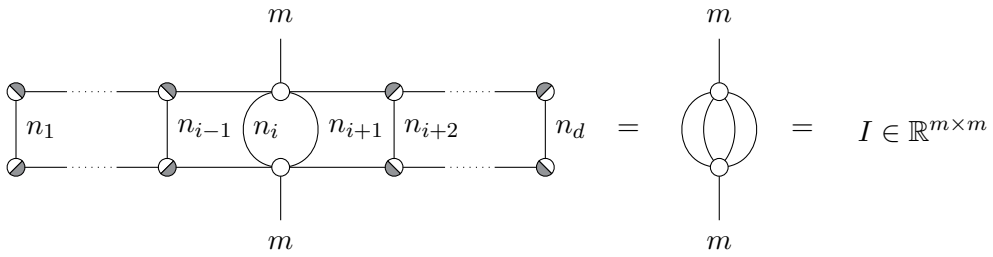
with $\mathbf{V} = \langle \mathbf{I}^{(i,i+1)}, v \rangle_m \in \mathbb{R}^{r_{i-1} \times n_i \times n_{i+1} \times r_{i+1}}$ being a super-core containing the modes n_i and n_{i+1}. By decomposing this core, i.e. applying a QR factorization or SVD to $\mathbf{V}\Big|^{n_{i+1},r_{i+1}}_{r_{i-1},n_i}$ and folding the resulting matrices, we obtain two TT cores $\mathbf{V}_1 \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$ and $\mathbf{V}_2 \in \mathbb{R}^{r_i \times n_{i+1} \times r_{i+1}}$, where r_i is the new TT rank given by the (truncated) decomposition, such that

$$\mathbf{Q}_{i,i+1} \cdot v = \left[\mathbf{T}^{(i)}\right] \otimes \cdots \otimes \left[\mathbf{T}^{(i-1)}\right] \otimes \left[\mathbf{V}_1\right] \otimes \left[\mathbf{V}_2\right] \otimes \left[\mathbf{T}^{(i+2)}\right] \otimes \cdots \otimes \left[\mathbf{T}^{(d)}\right].$$

With the aid of the retraction operators, we are able to construct a series of reduced systems of linear equations by fixing all cores except one (ALS) or two (MALS), respectively. The solution of the lower-dimensional system then represents the vectorization of the optimized TT core. However, before we go into the details in the next section, we show that the retraction operators $\mathbf{Q}_i$ and $\mathbf{Q}_{i,i+1}$ are orthonormal under the assumption that the cores have an orthonormal structure, cf. [37]. Similar to the notations (4.2.5) and (4.2.7) for the multiplication of the retraction operators with a vector, we introduce the notations $\mathbf{Q}_i^T \cdot \mathbf{Q}_i$ and $\mathbf{Q}_{i,i+1}^T \cdot \mathbf{Q}_{i,i+1}$, respectively, for the contraction of the operators with themselves, cf. Figure 4.3.



(a)



(b)

Figure 4.3: *Orthonormality of the retraction operators: (a) Contraction of* $\mathbf{Q}_i$ *with itself. Since left- and right-orthonormal cores cancel out, the result is the identity matrix in* $\mathbb{R}^{m\times m}$ *with* $m = r_{i-1} \cdot n_i \cdot r_i$*. (b) Contraction of* $\mathbf{Q}_{i,i+1}$ *with itself, the result is the identity matrix in* $\mathbb{R}^{m\times m}$ *with* $m = r_{i-1} \cdot n_i \cdot n_{i+1} \cdot r_{i+1}$*.*

Theorem 4.2.4. *Provided that the TT cores* $\mathbf{T}^{(j)}$ *are left-orthonormal for* $j = 1, \ldots, i-1$ *and right-orthonormal for* $j = i+1, \ldots, d$ *(*$j = i+2, \ldots, d$ *for MALS), the retraction operators* $\mathbf{Q}_i$ *and* $\mathbf{Q}_{i,i+1}$ *satisfy*

$$\mathbf{Q}_i^T \cdot \mathbf{Q}_i = I \in \mathbb{R}^{(r_{i-1}\cdot n_i \cdot r_i)\times(r_{i-1}\cdot n_i \cdot r_i)},$$

and

$$\mathbf{Q}_{i,i+1}^T \cdot \mathbf{Q}_{i,i+1} = I \in \mathbb{R}^{(r_{i-1}\cdot n_i \cdot n_{i+1} \cdot r_{i+1})\times(r_{i-1}\cdot n_i \cdot n_{i+1} \cdot r_{i+1})},$$

where I *denotes the identity matrix in the respective space.*

Proof. As illustrated in Figure 4.3, the contraction of the retraction operators yields a sequence of identity matrices, cf. Figure 3.7, leading to the contraction of $\mathbf{I}^{(i)}$ and $\mathbf{I}^{(i,i+1)}$ with themselves. It follows from the definitions (4.2.4) and (4.2.6) that the results are identity matrices. □

4.2.3. Computational Scheme

Instead of finding the minimizer of the functional J given in (4.2.3) in a single step, which may be infeasible for high-dimensional tensors, the idea of ALS and MALS,

respectively, is to optimize the TT cores of a given initial guess $\mathbf{T} \in \mathbb{R}^N$ successively. In order to do that, we consider the corresponding functionals of the form

$$(J \circ \mathbf{Q}_i)(v) = \frac{1}{2} v^T \mathbf{Q}_i^T \mathbf{A} \mathbf{Q}_i v - v^T \mathbf{Q}_i^T \mathbf{U}, \tag{4.2.8}$$

or, in the MALS case,

$$(J \circ \mathbf{Q}_{i,i+1})(w) = \frac{1}{2} w^T \mathbf{Q}_{i,i+1}^T \mathbf{A} \mathbf{Q}_{i,i+1} w - w^T \mathbf{Q}_{i,i+1}^T \mathbf{U}, \tag{4.2.9}$$

where $\mathbf{Q}_i$ and $\mathbf{Q}_{i,i+1}$ are the retraction operators as defined above. The arguments v and w are vectors of suitable length, i.e. $v \in \mathbb{R}^{r_{i-1} \cdot n_i \cdot r_i}$ and $w \in \mathbb{R}^{r_{i-1} \cdot n_i \cdot n_{i+1} \cdot r_{i+1}}$. For stationary points, the gradients of (4.2.8) and (4.2.9), respectively, are equal to zero, i.e.

$$\begin{aligned} \nabla(J \circ \mathbf{Q}_i)(v) &= \mathbf{Q}_i^T \mathbf{A} \mathbf{Q}_i v - \mathbf{Q}_i^T \mathbf{U} = 0 \in \mathbb{R}^{r_{i-1} \cdot n_i \cdot r_i}, \\ \nabla(J \circ \mathbf{Q}_{i,i+1})(w) &= \mathbf{Q}_{i,i+1}^T \mathbf{A} \mathbf{Q}_{i,i+1} w - \mathbf{Q}_{i,i+1}^T \mathbf{U} = 0 \in \mathbb{R}^{r_{i-1} \cdot n_i \cdot n_{i+1} \cdot r_{i+1}}. \end{aligned}$$

Thus, we can solve a lower-dimensional classical system of linear equations in order to optimize a single TT core. Defining $A_i = \mathbf{Q}_i^T \mathbf{A} \mathbf{Q}_i$, $u_i = \mathbf{Q}_i^T \mathbf{U}$, $A_{i,i+1} = \mathbf{Q}_{i,i+1}^T \mathbf{A} \mathbf{Q}_{i,i+1}$, and $u_{i,i+1} = \mathbf{Q}_i^T \mathbf{U}$, the optimization problems in each iteration step correspond to solving

$$A_i v = u_i, \quad i = 1, \ldots, d, \tag{4.2.10}$$

and

$$A_{i,i+1} w = u_{i,i+1}, \quad i = 1, \ldots, d-1, \tag{4.2.11}$$

respectively. As shown in [37], the matrices A_i and $A_{i,i+1}$ are symmetric positive definite.

Theorem 4.2.5. *Assuming that the fixed TT cores satisfy the properties described in Theorem 4.2.4, the systems of linear equations of the form* (4.2.10) *and* (4.2.11)*, respectively, have a unique solution.*

Proof. Since the TT operator $\mathbf{A}$ is symmetric, it holds that $A_i^T = A_i$ and $A_{i,i+1}^T = A_{i,i+1}$. Furthermore, $\mathbf{Q}_i$ and $\mathbf{Q}_{i,i+1}$ are injective with left inverses $\mathbf{Q}_i^T$ and $\mathbf{Q}_{i,i+1}^T$, respectively. Therefore, $\mathbf{Q}_i v = 0$ ($\mathbf{Q}_{i,i+1} w = 0$) if and only if $v = 0$ ($w = 0$). This and property (ii) from (4.2.2) imply that $v^T A_i v > 0$ for $v \neq 0$ and $w^T A_{i,i+1} w > 0$ for $w \neq 0$. Thus, the matrices A_i and $A_{i,i+1}$ are symmetric positive definite and (4.2.10) and (4.2.11) have a unique solution. □

Given a right-orthonormal tensor train $\mathbf{T}$, the basic idea of ALS and MALS, respectively, is to optimize the cores $\mathbf{T}^{(1)}$ to $\mathbf{T}^{(d)}$ (first half sweep) ensuring that the corresponding retraction operators satisfy the properties described in Theorem 4.2.4 in every iteration step. Afterwards, the cores are optimized in reverse order (second half sweep). The result is then again a right-orthonormal tensor train approximating the solution of (4.2.1). This procedure can also be repeated in order to increase the accuracy of the solution. Adaption to left-orthonormal tensor trains is also possible. Here, we describe both computational schemes using the graphical representation of tensor trains, see Figure 4.4 and Figure 4.5. A consideration from an algorithmic point of view is given in the next section. Further properties of the ALS and MALS algorithms for systems of linear equations will be examined in Section 4.4.

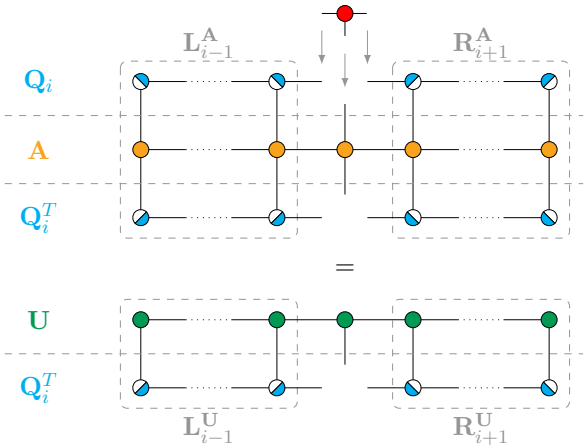


Figure 4.4: *Illustration of ALS: The tensor* $\mathbf{I}^{(i)}$ *is omitted for the sake of simplicity. The TT operator* $\mathbf{A}$ *(depicted by orange circles) is contracted with the retraction operator* $\mathbf{Q}_i$ *(depicted by blue circles) from both sides. Contracting the tensor train* $\mathbf{U}$ *(depicted by green circles) with the retraction operator provides the right-hand side. The aim is to find an optimized tensor core (red circle) such that the contraction with* $\mathbf{Q}_i^T\mathbf{A}\mathbf{Q}_i$ *yields* $\mathbf{Q}_i^T\mathbf{U}$*. The tensors* $\mathbf{L}^{\mathbf{A}}_{i-1}$ *and* $\mathbf{R}^{\mathbf{A}}_{i+1}$ *can be used for the evaluation of the left-hand side.*

4.2.4. Algorithmic Aspects

In order to ensure the left- and right-orthonormality, respectively, of the fixed TT cores in every iteration step (and therefore the stability of the method), (M)ALS decomposes the updated TT core $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{i-1}\times n_i\times r_i}$, which is obtained by solving (4.2.10) and folding the solution, see [37]. That is, during the first half sweep of ALS, we apply a full QR decomposition and only keep the left-orthonormal part, i.e. after computing the decomposition of the left-unfolding (3.4.5)

$$\mathcal{L}\left(\mathbf{T}^{(i)}\right) = Q \cdot R, \tag{4.2.12}$$

with $Q \in \mathbb{R}^{(r_{i-1}\cdot n_i)\times r_i}$ and $R \in \mathbb{R}^{r_i\times r_i}$, we set $\mathbf{T}^{(i)}$ to a reshaped version of Q such that $\mathcal{L}\left(\mathbf{T}^{(i)}\right) = Q$. The non-orthonormal part is then shifted to the next core,

i.e. we compute $R \cdot \mathcal{R}\left(\mathbf{T}^{(i+1)}\right)$, where $\mathcal{R}\left(\mathbf{T}^{(i+1)}\right)$ denotes the right-unfolding of $\mathbf{T}^{(i+1)}$, and set the $(i+1)$-th core to the folding of the result. However, we can omit the latter calculation since the succeeding TT core is optimized in the next iteration step anyway.

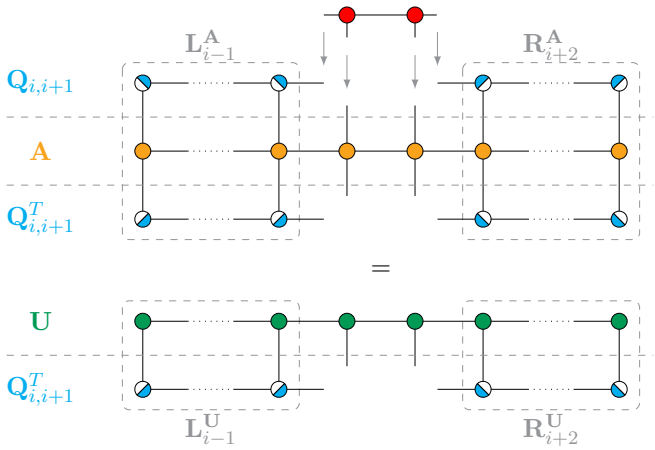


Figure 4.5: *Illustration of MALS: Again, the tensor* $\mathbf{I}^{(i,i+1)}$ *is omitted. This time,* $\mathbf{A}$ *(depicted by orange circles) and* $\mathbf{U}$ *(depicted by green circles) are contracted with the retraction operator* $\mathbf{Q}_{i,i+1}$ *(depicted by blue circles). The aim is to find an optimized super-core (depicted by red circles) which provides, after decomposing, the updated cores* $\mathbf{T}^{(i)}$ *and* $\mathbf{T}^{(i+1)}$*. The tensors* $\mathbf{L}^{\mathbf{A}}_{i-1}$ *and* $\mathbf{R}^{\mathbf{A}}_{i+2}$ *can be used for the evaluation of the left-hand side.*

During the back sweep of ALS, the orthonormalization is adapted to the inverted direction. Now, we compute

$$\left(\mathcal{R}\left(\mathbf{T}^{(i)}\right)\right)^T = Q \cdot R \qquad \Leftrightarrow \qquad \mathcal{R}\left(\mathbf{T}^{(i)}\right) = R^T \cdot Q^T, \tag{4.2.13}$$

with $Q \in \mathbb{R}^{(n_i \cdot r_i) \times r_{i-1}}$ and $R \in \mathbb{R}^{r_{i-1} \times r_{i-1}}$, and set $\mathbf{T}^{(i)}$ to a reshaped version of Q^T. Again, we can omit the contraction of the matrix R^T with the core $\mathbf{T}^{(i-1)}$. Eventually, when optimizing the first core at the end of the second half sweep, $\mathbf{T}^{(1)}$ is set to the folded solution of (4.2.10), without subsequent decomposition.

Considering the optimization of $\mathbf{T}^{(i)}$ and $\mathbf{T}^{(i+1)}$ by employing MALS, we only keep the core $\mathbf{T}^{(i)}$ during the first half sweep and the core $\mathbf{T}^{(i+1)}$ during the second half sweep. We apply a compact SVD of the solution w of (4.2.11), i.e. we reshape $w \in \mathbb{R}^{r_{i-1} \cdot n_i \cdot n_{i+1} \cdot r_{i+1}}$ into $W \in \mathbb{R}^{(r_{i-1} \cdot n_i) \times (n_{i+1} \cdot r_{i+1})}$ and compute

$$W = U \cdot \Sigma \cdot V^T, \tag{4.2.14}$$

with $U \in \mathbb{R}^{(r_{i-1} \cdot n_i) \times s}$, $\Sigma \in \mathbb{R}^{s \times s}$, and $V \in \mathbb{R}^{(n_{i+1} \cdot r_{i+1}) \times s}$. Note that Σ only contains the non-zero singular values of W such that $s \leq \min\{r_{i-1} \cdot n_i, n_{i+1} \cdot r_{i+1}\}$. Furthermore, we also consider a truncation of the SVD (4.2.14) in order to control the TT ranks of the tensor train $\mathbf{T}$. For increasing i, we set the TT core $\mathbf{T}^{(i)}$ to a reshaped

version of U. During the back sweep, the core $\mathbf{T}^{(i+1)}$ is replaced by a folding of V^T. In the last iteration step of the second half sweep (solve $A_{1,2}\, v = u_{1,2}$), we additionally set $\mathbf{T}^{(1)}$ to a reshaped version of $U \cdot \Sigma$ such that the output of MALS is again a right-orthonormal tensor train.

The algorithms iteratively compute the tensors needed for the construction of the systems of linear equations (4.2.10) and (4.2.11), respectively. Considering Figure 4.4, the matrix A_i can be seen as a contraction of a tensor $\mathbf{L}^{\mathbf{A}}_{i-1}$ of order 3 from the left and a tensor $\mathbf{R}^{\mathbf{A}}_{i+1}$, also of order 3, from the right with the TT core $\mathbf{A}^{(i)}$ of the operator. The matrix A_i is then the matricization of the result, where corresponding free modes are joined in order to form the row and column indices, i.e.

$$A_i = \left\langle \left\langle \mathbf{L}^{\mathbf{A}}_{i-1}, \mathbf{A}^{(i)} \right\rangle_{R_{i-1}}, \mathbf{R}^{\mathbf{A}}_{i+1} \right\rangle_{R_i} \Bigg|^{\tilde{r}_{i-1},\tilde{n}_i,\tilde{r}_i}_{r_{i-1},n_i,r_i}, \tag{4.2.15}$$

with R_{i-1} and R_i being the rank indices of $\mathbf{A}^{(i)}$ and r_{i-1} and r_i being the rank indices of $\mathbf{T}^{(i)}$. The quantities $\tilde{r}_{i-1}$, $\tilde{n}_i$, and $\tilde{r}_i$ are introduced for a better overview. They represent the upper free modes from $\mathbf{P}_i^T \mathbf{A} \mathbf{Q}_i$ in Figure 4.4. For the tensors $\mathbf{L}^{\mathbf{A}}_i$, $i = 0, \dots, d-1$, and $\mathbf{R}^{\mathbf{A}}_i$, $i = 2, \dots, d+1$, we define $\mathbf{L}^{\mathbf{A}}_0 = 1$ and $\mathbf{R}^{\mathbf{A}}_{d+1} = 1$. Then, the preceding/succeeding tensors can be computed by the recursive formulae

$$\mathbf{L}^{\mathbf{A}}_i = \left\langle \left\langle \left\langle \mathbf{L}^{\mathbf{A}}_{i-1}, \mathbf{T}^{(i)} \right\rangle_{r_{i-1}}, \mathbf{A}^{(i)} \right\rangle_{R_{i-1},n_i}, \mathbf{T}^{(i)} \right\rangle_{\tilde{r}_{i-1},\tilde{n}_i} \quad \text{for } 1 \le i \le d-1, \tag{4.2.16}$$

and

$$\mathbf{R}^{\mathbf{A}}_i = \left\langle \left\langle \left\langle \mathbf{R}^{\mathbf{A}}_{i+1}, \mathbf{T}^{(i)} \right\rangle_{r_i}, \mathbf{A}^{(i)} \right\rangle_{R_i,n_i}, \mathbf{T}^{(i)} \right\rangle_{\tilde{r}_i,\tilde{n}_i} \quad \text{for } 2 \le i \le d, \tag{4.2.17}$$

respectively. Analogously to (4.2.15), u_i is the contraction of $\mathbf{U}^{(i)} \in \mathbb{R}^{s_{i-1} \times n_i \times s_i}$ with recursively defined order-2 tensors $\mathbf{L}^{\mathbf{U}}_{i-1}$ and $\mathbf{R}^{\mathbf{U}}_{i+1}$ ($\mathbf{L}^{\mathbf{U}}_0 = \mathbf{R}^{\mathbf{U}}_{d+1} = 1$), i.e.

$$u_i = \left\langle \left\langle \mathbf{L}^{\mathbf{U}}_{i-1}, \mathbf{U}^{(i)} \right\rangle_{s_{i-1}}, \mathbf{R}^{\mathbf{U}}_{i+1} \right\rangle_{s_i} \Bigg|_{r_{i-1},n_i,r_i},$$

where

$$\mathbf{L}^{\mathbf{U}}_i = \left\langle \left\langle \mathbf{L}^{\mathbf{U}}_{i-1}, \mathbf{T}^{(i)} \right\rangle_{r_{i-1}}, \mathbf{U}^{(i)} \right\rangle_{s_{i-1},n_i} \quad \text{for } 1 \le i \le d-1, \tag{4.2.18}$$

and

$$\mathbf{R}_i^{\mathbf{U}} = \left\langle \left\langle \mathbf{R}_{i+1}^{\mathbf{U}}, \mathbf{T}^{(i)} \right\rangle_{r_i}, \mathbf{U}^{(i)} \right\rangle_{s_i,n_i} \quad \text{for } 2 \leq i \leq d. \tag{4.2.19}$$

This is also shown in Figure 4.4. The computations (4.2.16), (4.2.17), (4.2.18), and (4.2.19) can be done, for instance, in MATLAB using the functions `permute` and `reshape`. However, the matrix A_i does not have to be computed explicitly. Applying iterative solvers for systems of linear equations with symmetric positive definite matrices, see [64], we can reduce the computational effort for evaluating $A_i\, v$ significantly, cf. [37]. Furthermore, the quantities $\mathbf{R}_i^{\mathbf{A}}$ and $\mathbf{R}_i^{\mathbf{U}}$ for $i = d, \ldots, 2$ can be precomputed such that we only have to construct the tensors $\mathbf{L}_i^{\mathbf{A}}$ and $\mathbf{L}_i^{\mathbf{U}}$ during the first half sweep. For the second half sweep, we then keep $\mathbf{L}_i^{\mathbf{A}}$ and $\mathbf{L}_i^{\mathbf{U}}$ for $i = 1, \ldots, d-1$ and compute the tensors $\mathbf{R}_i^{\mathbf{A}}$ and $\mathbf{R}_i^{\mathbf{U}}$ for a certain number i at each iteration step.

The systems of linear equations (4.2.11) corresponding to MALS – if computed explicitly – are given by

$$A_{i,i+1} = \left\langle \left\langle \left\langle \mathbf{L}_{i-1}^{\mathbf{A}}, \mathbf{A}^{(i)} \right\rangle_{R_{i-1}}, \mathbf{A}^{(i+1)} \right\rangle_{R_i}, \mathbf{R}_{i+2}^{\mathbf{A}} \right\rangle_{R_{i+1}} \Bigg|_{r_{i-1},n_i,n_{i+1},r_i}^{\tilde{r}_{i-1},\tilde{n}_i,\tilde{n}_{i+1},\tilde{r}_i},$$

and

$$u_{i,i+1} = \left\langle \left\langle \left\langle \mathbf{L}_{i-1}^{\mathbf{U}}, \mathbf{U}^{(i)} \right\rangle_{s_{i-1}}, \mathbf{U}^{(i+1)} \right\rangle_{s_i}, \mathbf{R}_{i+2}^{\mathbf{U}} \right\rangle_{s_{i+1}} \Bigg|_{r_{i-1},n_i,n_{i+1},r_i},$$

for $i = 1, \ldots, d-1$, see Figure 4.5. The pseudocodes of ALS and MALS are presented in Algorithm 11 and in Algorithm 12, respectively, see Appendix A.2.

4.3. (M)ALS for Eigenvalue Problems

After describing (M)ALS for systems of linear equations, we now want to consider eigenvalue problems in the TT format. As we will see, the basic procedures of ALS and MALS for eigenvalue problems are similar to the ones for systems of linear equations. The main difference is the type of optimization problem which has to be solved in the iteration steps.

4.3.1. Problem Statement

We consider an eigenvalue problem

$$\mathbf{A} \cdot \mathbf{T} = \lambda \cdot \mathbf{T}, \quad \mathbf{T} \neq \mathbf{0} \tag{4.3.1}$$

which is given in the TT format with tensor trains $\mathbf{A} \in \mathbb{R}^{N \times N}$ and $\mathbf{T} \in \mathbb{R}^N$. Note that also $\mathbf{0}$ is a tensor in $\mathbb{R}^N$. The scalar $\lambda \in \mathbb{R}$ is then an eigenvalue of $\mathbf{A}$ and it holds that $\mathrm{mat}(\mathbf{A}) \cdot \mathrm{vec}(\mathbf{T}) = \lambda \cdot \mathrm{vec}(\mathbf{T})$. Again, we assume that the TT operator $\mathbf{A}$ is symmetric, but not necessarily positive definite. In order to find solutions of (4.3.1), it was shown in [37] that we can consider a functional J which is given by the *Rayleigh quotient* [75].

Theorem 4.3.1. *The solutions of the eigenvalue problems* (4.3.1) *are the stationary points of the functional J given by*

$$J(\mathbf{T}) = \frac{1}{2} \frac{\mathbf{T}^T \mathbf{A} \mathbf{T}}{\mathbf{T}^T \mathbf{T}}. \tag{4.3.2}$$

Proof. As in the proof of Theorem 4.2.1, we can reduce (4.3.1) to a standard eigenvalue problem $Av = \lambda v$ with a symmetric matrix A and a vector v. Thus, we consider the functional $J(v) = (v^T A v)/(2v^T v)$ with

$$\nabla J(v) = \frac{\|v\|_2^2 \cdot Av - (v^T A v) \cdot v}{\|v\|_2^4}.$$

For stationary points, it then holds that

$$Av = \frac{v^T A v}{v^T v} v,$$

where the fraction on the right-hand side is exactly the Rayleigh quotient of A and v. We see that any stationary point v has to be an eigenvector of the matrix A. Assuming the corresponding eigenvalue is λ_v, we indeed obtain

$$Av = \frac{v^T \lambda_v v}{v^T v} v = \lambda_v v. \qquad \square$$

4.3.2. Computational Scheme

By following Theorem 4.3.1, we will focus on finding stationary points of the functional J instead of considering the eigenvalue problem (4.3.1). Similar to the treatment of systems of linear equations, we want to optimize the TT cores of an initial guess iteratively by considering the function composition

$$(J \circ \mathbf{Q}_i)(v) = \frac{1}{2} \frac{v^T \mathbf{Q}_i^T \mathbf{A} \mathbf{Q}_i v}{v^T \mathbf{Q}_i^T \mathbf{Q}_i v}.$$

Assuming that the fixed TT cores satisfy the properties described in Theorem 4.2.4, it holds that

$$\nabla(J \circ \mathbf{Q}_i)(v) = 0 \qquad \Leftrightarrow \qquad A_i v = \frac{v^T A_i v}{v^T v} v,$$

where $A_i = \mathbf{Q}_i^T \mathbf{A} \mathbf{Q}_i$ and the fraction on the right-hand side is the Rayleigh quotient of A_i and v. Thus, in order to compute the smallest or largest eigenvalue of the TT operator $\mathbf{A}$, we minimize or maximize, respectively, the functional $J \circ \mathbf{Q}_i$ which is the same as finding the smallest/largest eigenvalue of A_i.

Additionally, we utilize the BTT format, see Section 3.5.2. As presented in [19], it is possible to approximate the b smallest and largest eigenvalues, respectively, of the operator $\mathbf{A}$ in one step. That is, we compute the b smallest/largest eigenvectors of A_i and then adapt the orthonormalization procedure of ALS such that the extra index is shifted to the next core. Since a tensor in BTT format represents different tensor trains by replacing only one core, it is extremely unlikely to be able to accurately approximate a large number of eigentensors at once. However, as we will see in Chapter 12, one can find examples where it is possible to compute the most interesting eigenvalues and eigentensors of a TT operator simultaneously.

For MALS, we compute the extremal eigenvalues of the matrix $A_{i,i+1}$ and then split the resulting tensor in an orthonormal part and a part carrying the extra index b. As for systems of linear equations, we rely on iterative solvers for symmetric eigenvalue problems, see [64]. Again, cf. Section 4.2.4, we combine the basic computational scheme of ALS and MALS, respectively, with the recursive computation of the tensors $\mathbf{L}_i^{\mathbf{A}}$, $\mathbf{R}_i^{\mathbf{A}}$ and an orthonormalization procedure which also shifts the BTT mode b. The resulting methods are shown in Algorithm 13 and 14, respectively, see Appendix A.2.

4.4. Properties of (M)ALS

At the end of this chapter, we want to mention important properties of (M)ALS including computational aspects and convergence properties. Table 4.1 shows the computational complexities of Algorithms 11-14, see Appendix A.2. Here, we denote the maximum TT rank of $\mathbf{T}$ and $\mathbf{U}$ by r, the maximum TT rank of $\mathbf{A}$ by R, and the maximum mode size by n.

For solving systems of linear equations as well as for the calculation of eigenvalues, the computations of the essential tensors (i.e. $\mathbf{L}_i^A$, $\mathbf{L}_i^U$, $\mathbf{R}_i^A$, $\mathbf{R}_i^U$) can be estimated by $O(d\, r^3\, R^2\, n^2)$. Applying iterative solvers for the low-dimensional subproblems, e.g. conjugate gradient method or power iteration [64], the dominant operations are the evaluations of $A_i\, v$ and $A_{i,i+1}\, v$, respectively. This can be done in $O(r^3\, R^2\, n^2)$ (ALS) and $O(r^3\, R^2\, n^3)$ (MALS), respectively, contracting the corresponding tensors in a suitable way, cf. [37]. Multiplication by d and the number of maximum iterations γ of the inner solvers yields the estimations for (M)ALS for systems of linear equations. Considering eigenvalue problems, we obtain the same complexities for the approximation of one eigenpair. Thus, the computation of the b

Table 4.1.: *Computational complexity of (M)ALS: The number d denotes the order of the involved tensors, n the maximum of all mode sizes, r and R the maximum TT ranks of the tensor trains in $\mathbb{R}^N$ and of the TT operator in $\mathbb{R}^{N\times N}$, respectively, γ the number of iterations of iterative solvers for the low-dimensional systems, and b the number of eigenvalues and eigentensors to be computed using the (M)ALS algorithms.*

Algorithm	Complexity
ALS for systems of linear equations	$O(\gamma\, d\, r^3\, R^2\, n^2)$
MALS for systems of linear equations	$O(\gamma\, d\, r^3\, R^2\, n^3)$
ALS for eigenvalue problems	$O(b\,\gamma\, d\, r^3\, R^2\, n^2)$
MALS for eigenvalue problems	$O(b\,\gamma\, d\, r^3\, R^2\, n^3)$

smallest/largest eigenvalues has the time complexities given in Table 4.1. Note that the QR-factorizations and SVDs implemented in the (M)ALS algorithms are also included in the estimations. Computing a QR factorization of a matrix $A \in \mathbb{R}^{k\times l}$, $k \geq l$ requires $O(k\, l^2)$, see [64], while an SVD requires $O(k\, l^2 + l^3)$, see [76]. Storing all involved tensors requires $O(d\, r^2\, R^2\, n^2)$.

As we have already shown, the reduced matrices A_i and $A_{i,i+1}$ are symmetric if the TT operator $\mathbf{A}$ itself is symmetric and, additionally, positive definite if $\mathbf{A}$ is positive definite. Let $\Lambda_{\min}$ and $\Lambda_{\max}$ denote the smallest and largest eigenvalue, respectively, of the operator $\mathbf{A}$. Due to the symmetry of $\mathbf{A}$, the extremal eigenvalues $\lambda_{\min}$, $\lambda_{\max}$ of the reduced matrices satisfy

$$\lambda_{\min} = \min_{\substack{v\in\mathbb{R}^m \\ v\neq 0}} \frac{v^T A\, v}{v^T\, v} \geq \min_{\substack{\mathbf{T}\in\mathbb{R}^N \\ \mathbf{T}\neq\mathbf{0}}} \frac{\mathbf{T}^T\, \mathbf{A}\, \mathbf{T}}{\mathbf{T}^T\, \mathbf{T}} = \Lambda_{\min},$$

and

$$\lambda_{\max} = \max_{\substack{v\in\mathbb{R}^m \\ v\neq 0}} \frac{v^T A\, v}{v^T\, v} \leq \max_{\substack{\mathbf{T}\in\mathbb{R}^N \\ \mathbf{T}\neq\mathbf{0}}} \frac{\mathbf{T}^T\, \mathbf{A}\, \mathbf{T}}{\mathbf{T}^T\, \mathbf{T}} = \Lambda_{\max},$$

with $A = A_i$, $m = r_{i-1}\cdot n_i \cdot r_i$ and $A = A_{i,i+1}$, $m = r_{i-1}\cdot n_i \cdot n_{i+1} \cdot r_{i+1}$, respectively. If $\mathbf{A}$ is also positive definite, all eigenvalues of $\mathbf{A}$ and A are positive. As shown in [37], we obtain for the condition numbers

$$\operatorname{cond}(A) = \frac{\lambda_{\max}}{\lambda_{\min}} \leq \frac{\Lambda_{\max}}{\Lambda_{\min}} = \operatorname{cond}(\mathbf{A}).$$

That is, for any optimization step of (M)ALS, the condition number of A_i and $A_{i,i+1}$, respectively, is bounded by the condition number of the given TT operator.

For systems of linear equations, if the given TT operator $\mathbf{A} \in \mathbb{R}^{N\times N}$ is not sym-

metric positive definite, we could consider the operator $\mathbf{A}^T\mathbf{A}$, which is the counterpart to a *Gram matrix*. The solution of the system $\mathbf{A}\cdot\mathbf{T} = \mathbf{U}$ is also the solution of the *normal equation* $(\mathbf{A}^T\mathbf{A})\cdot\mathbf{T} = \mathbf{A}^T\mathbf{U}$ with $\mathbf{A}^T\mathbf{A}$ being symmetric positive definite. However, as we explained in Section 3.4.2, the TT ranks are squared due to the multiplication of two tensor trains. That is, the treatment of $\mathbf{A}^T\mathbf{A}$ may be much more expensive in terms of computational complexity and storage consumption than the treatment of $\mathbf{A}$. Additionally, it holds that $\mathrm{cond}(\mathbf{A}^T\mathbf{A}) = \mathrm{cond}(\mathbf{A})^2$.

In [37], it was shown that ALS and MALS for systems of linear equations and eigenvalue problems are monotonic in the sense that $J(\mathbf{U}) \leq J(\mathbf{T})$, where J is one of the functionals (4.2.3), (4.3.2) and $\mathbf{U}$ is the result of (M)ALS after one half sweep with initial guess $\mathbf{T}$. But, even though it is clear that the treatment of the functionals is equivalent to solving the original problems, we cannot ensure the convergence of ALS to the solution (or a low-rank TT approximation of it) in any case. We have to include the possible existence of local minima/maxima of the parametrized problems – even if the minimum/maximum of the functional J is unique. However, extensive numerical tests and practical experience show a remarkable convergence behavior of (M)ALS. Furthermore, there are cases where the convergence of (M)ALS can be shown. Under several assumptions, the local linear convergence of ALS for convex functionals was proven in [77]. Additionally, one can ensure the convergence of ALS to an exact TT decomposition under certain conditions if the algorithm is used for the approximation of a given tensor $\mathbf{T}$ with known TT ranks, see [37].

4.5. Methods for Solving Initial Value Problems

The algorithms introduced in the previous sections will be later employed for the purpose of solving initial value problems, in particular *ordinary differential equations* (ODEs) given by a *Markovian master equation*, see Section 5, together with an initial probability distribution $\mathbf{P}_0 \in \mathbb{R}^N$ at time $t_0 = 0$. In order to compute time-dependent or stationary distributions, we will use implicit integration schemes such as the implicit Euler method [78] or the trapezoidal rule [79]. The reason for this is that the calculation of the distribution $\mathbf{P}_{k+1}$ at time t_{k+1} from $\mathbf{P}_k$ at time t_k requires at least one multiplication of a TT operator with a tensor train when applying explicit methods to linear ODEs. As mentioned in Section 3.4.2, the product has TT ranks equal to the products of the corresponding ranks of both tensors. Thus, after each iteration, we would have to truncate the ranks of the resulting tensor $\mathbf{P}_{k+1}$ in order to keep the computations feasible, e.g. by using Algorithm 10 in combination with truncated SVDs.

For an implicit method, on the other hand, we have to solve a system of linear equations in the TT format at every iteration step. One of the most basic implicit schemes is the implicit Euler method, which, for a linear ODE

$$\frac{d}{dt}\mathbf{P}(t) = \mathbf{A}\mathbf{P}(t), \tag{4.5.1}$$

with $\mathbf{A} \in \mathbb{R}^{N \times N}$ (in TT format), requires to solve systems of linear equations of the form

$$(\mathbf{I} - \tau \mathbf{A})\mathbf{P}_k = \mathbf{P}_{k-1}, \quad k > 0, \tag{4.5.2}$$

where $\tau \in \mathbb{R}^+$ denotes the step size and $\mathbf{I} \in \mathbb{R}^{N \times N}$ is the identity tensor, see Example 2.1.1. For later considerations, we define the residual error $e_{\mathrm{IE},k}$ as

$$e_{\mathrm{IE},k} = \frac{\|(\mathbf{I} - \tau \mathbf{A})\mathbf{P}_k - \mathbf{P}_{k-1}\|_2}{\|\mathbf{P}_{k-1}\|_2}, \tag{4.5.3}$$

where $\| \, . \, \|_2$ denotes the 2-norm considered in Section 2.5 and Section 3.4.4. Using an integration scheme of higher order, we will also apply the second-order trapezoidal rule for transient processes. The systems of linear equations in each integration step then become

$$\left(\mathbf{I} - \frac{\tau}{2}\mathbf{A}\right)\mathbf{P}_k = \left(\mathbf{I} + \frac{\tau}{2}\mathbf{A}\right)\mathbf{P}_{k-1}, \quad k > 0. \tag{4.5.4}$$

In this case, the residual error for the evaluation of the accuracy of our results is given by

$$e_{\mathrm{TR},k} = \frac{\left\|\left(\mathbf{I} - \frac{\tau}{2}\mathbf{A}\right)\mathbf{P}_k - \left(\mathbf{I} + \frac{\tau}{2}\mathbf{A}\right)\mathbf{P}_{k-1}\right\|_2}{\left\|\left(\mathbf{I} + \frac{\tau}{2}\mathbf{A}\right)\mathbf{P}_{k-1}\right\|_2}. \tag{4.5.5}$$

In order to approximate the solutions of (4.5.2) and (4.5.4), ALS and MALS, respectively, are used. In addition to the fact that implicit integration schemes are more suitable for the solution of stiff equations than explicit methods, we have control over the ranks of the TT approximations without implementing computationally expensive tensor multiplications and subsequent rank truncations.

If the ODE (4.5.1) converges to a unique stationary distribution, we can reformulate the problem of computing this distribution as an eigenvalue problem. That is, we consider the problem

$$(\mathbf{I} + \mathbf{A})\,\mathbf{P} = \lambda \mathbf{P}, \quad \|\mathbf{P}\|_1 = 1, \tag{4.5.6}$$

using the methods presented in Section 4.3.

Part II

Progress in Tensor-Train Decompositions

"It is the curse of dimensionality, a malediction that has plagued the scientist from the earliest days."

Richard Bellman,
Adaptive Control Processes: A Guided Tour

In Part II of this thesis, we will present our own contribution to the concept of tensor-train decompositions, see [6, 9, 14]. After proposing a method to reformulate Markovian master equations in a tensor-based notation in Chapter 5, we will introduce SLIM decompositions corresponding to nearest-neighbor interaction systems in Chapter 6. Furthermore, in Chapter 7, we will describe how to compute pseudoinverses of certain matricizations of a tensor and present a tensor-based version of the dynamic mode decomposition. In Chapter 8, we will show how to compute a finite-dimensional approximation of the Perron–Frobenius operator in the tensor-train format.

5

Tensor Representation of Markovian Master Equations

In this chapter, we will explain how to derive tensor representations of Markovian master equations [48] corresponding to potentially high-dimensional systems. A special type of Markovian master equation describing the time-evolution of chemical systems, the chemical master equations [80], was already considered in a tensor-based context, e.g. in [4, 5]. Here, we will extend this method in order to express arbitrary Markovian master equations in a tensor notation.

5.1. Markov Jump Processes

Let us first recapitulate the definition of a continuous-time Markov process on a finite state space $\mathcal{S}$. A so-called *Markov jump process* is a stochastic process $\{\mathcal{X}(t) : t \in \mathbb{R}_0^+\}$, see e.g. [81], that has the Markov property [47]

$$\begin{aligned}\mathbb{P}(\mathcal{X}(t_{k+1}) = X_{k+1} \,|\, \mathcal{X}(t_k) = X_k \wedge \mathcal{X}(t_{k-1}) = X_{k-1} \wedge \cdots \wedge \mathcal{X}(t_0) = X_0) \\ = \mathbb{P}(\mathcal{X}(t_{k+1}) = X_{k+1} \,|\, \mathcal{X}(t_k) = X_k),\end{aligned} \tag{5.1.1}$$

for any finite set $0 \leq t_0 < t_1 < \cdots < t_k < t_{k+1}$ of times and corresponding set $X_0, X_1, \ldots, X_{k+1}$ of states in $\mathcal{S}$. Here, $\mathbb{P}$ denotes the probability measure corresponding to the probability space of the stochastic process. In particular, we will consider *homogeneous* Markov jump processes where the right-hand side of (5.1.1) only depends on $\delta t = t_{k+1} - t_k$.

Now, let $P(X,t) = \mathbb{P}(\mathcal{X}(t) = X)$ denote the probability that the system is in state $X \in \mathcal{S}$ at time t under the condition that it was in state X_0 at time t_0. For the sake of simplicity, the dependence on the initial state is omitted. The probability distribution $P(X,t)$ then obeys a *Markovian Master Equation* (MME) [48], given by

$$\frac{\partial}{\partial t} P(X,t) = \sum_Y W(X|Y)P(Y,t) - \sum_Y W(Y|X)P(X,t), \tag{5.1.2}$$

where $W(Y|X)$ is the transition rate to go from state X to state Y, i.e.

$$W(Y|X) = \lim_{t \searrow 0} \frac{\mathbb{P}(\mathcal{X}(t) = Y | \mathcal{X}(0) = X) - \mathbb{P}(\mathcal{X}(0) = Y | \mathcal{X}(0) = X)}{t},$$

for all $X, Y \in \mathcal{S}$. If we consider a finite state space, i.e. we identify the states by the set of natural numbers $\mathcal{S} = \{1, \ldots, n\}$, $n \in \mathbb{N}$, we can express (5.1.2) as an ODE of the form

$$\frac{\partial}{\partial t} P(t) = W^T \cdot P(t), \tag{5.1.3}$$

with $P(t) = (P(1,t), \ldots, P(n,t))^T \in \mathbb{R}^n$ and $W = (W(y|x))_{x,y \in \mathcal{S}}$. The matrix $W \in \mathbb{R}^{n \times n}$ is called *rate matrix* or *infinitesimal generator* and it holds that $W(y|x) \geq 0$ for all $x, y \in \mathcal{S}$, $x \neq y$, and $\sum_{y \in \mathcal{S}} W(y|x) = 0$, see [82].

5.2. Tensor-Based Representation of Infinitesimal Generators

The reformulation (5.1.3) is only feasible if we consider a moderate number of states. The aim here is to obtain a tensor-based counterpart of (5.1.3) for Markov jump processes on potentially high-dimensional state spaces. To achieve this, we first rewrite (5.1.2) in the form of a *chemical master equation* (CME). The reason for that is, on the one hand, that we can later apply this reformulation directly to chemical reaction networks considered in Chapter 9 and, on the other hand, that more general Markov processes on high-dimensional state spaces can be expressed in this notation, see [6, 9] as well as Chapter 10. Furthermore, the theory for tensor-based reformulations of CMEs was already provided in, e.g., [5].

If $W(Y|X) \neq 0$, we say there is an event R_μ, $\mu \in \mathbb{N}$, that causes a transition from state X to Y, $Y \neq X$. We assume that the state space is given by

$$\mathcal{S} = \{1, \ldots, n_1\} \times \{1, \ldots, n_2\} \times \cdots \times \{1, \ldots, n_d\}, \tag{5.2.1}$$

such that a state $X \in \mathcal{S}$ is a vector $(x_1, \ldots, x_d)^T \in \mathbb{N}^d$. We denote the net changes in the state vector X caused by a single execution of the event R_μ by the vector $\xi_\mu \in \mathbb{Z}^d$, i.e.

$$\xi_\mu = Y - X = (y_1 - x_1, \ldots, y_d - x_d)^T. \tag{5.2.2}$$

Assuming that the event R_μ – defined by the vector ξ_μ – can be executed for different states, i.e. the transition rates for going from X to $X + \xi_\mu$ are non-zero for several

states $X \in \mathcal{S}$, we furthermore define the *event propensity* a_μ as

$$a_\mu(X) = W(X + \xi_\mu | X). \tag{5.2.3}$$

Note that a_μ is only non-zero if X and $X + \xi_\mu$ are both in $\mathcal{S}$ and X complies with the requirements that R_μ can be executed, otherwise we set $a_\mu(X) = 0$. Thus, summing over all possible events $\mathrm{R}_1, \ldots, \mathrm{R}_\mathcal{M}$, $\mathcal{M} \in \mathbb{N}$, we obtain

$$\frac{\partial}{\partial t} P(X,t) = \sum_{\mu=1}^{\mathcal{M}} a_\mu(X - \xi_\mu) P(X - \xi_\mu, t) - a_\mu(X) P(X,t). \tag{5.2.4}$$

Due to the summation in (5.2.4), we consider exactly all possible states from which X can be reached and all states that can be reached from X by a single execution of one of the events R_μ. In fact, equation (5.2.4) has the same structure as a CME. However, a state does not necessarily represent numbers of molecules as it is the case for classical CMEs, see Section 9.2.

In order to express (5.2.4) using tensors, we identify each propensity function $a_\mu : \mathcal{S} \to \mathbb{R}$ with a tensor $\mathbf{a}_\mu \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, i.e. for a state $X = (x_1, \ldots, x_d)^T \in \mathcal{S}$, we define

$$(\mathbf{a}_\mu)_{x_1,\ldots,x_d} = a_\mu(X), \tag{5.2.5}$$

for $\mu = 1, \ldots, \mathcal{M}$. Expressing the propensity tensors in the canonical format, we write

$$\mathbf{a}_\mu = \sum_{k=1}^{r_\mu} \left(\mathbf{a}_\mu^{(1)}\right)_{k,:} \otimes \cdots \otimes \left(\mathbf{a}_\mu^{(d)}\right)_{k,:}, \tag{5.2.6}$$

with cores $\mathbf{a}_\mu^{(i)} \in \mathbb{R}^{r_\mu \times n_i}$ and canonical rank r_μ. Additionally, we gather the probabilities $P(X,t)$ in a tensor $\mathbf{P}(t) \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ with

$$(\mathbf{P}(t))_{x_1,\ldots,x_d} = P(X,t).$$

Definition 5.2.1. *Let $G_i(k) \in \mathbb{R}^{n_i \times n_i}$ denote the shift matrix given by $(G_i(k))_{x,y} := \delta_{y-x,k}$, where $\delta_{y-x,k}$ represents the Kronecker delta. Then the* multi-dimensional shift operators *$\mathbf{G}_\mu$ and $\mathbf{G}_0$ are defined as*

$$\mathbf{G}_\mu = G_1(-\xi_\mu(1)) \otimes \cdots \otimes G_d(-\xi_\mu(d))$$

and

$$\mathbf{G}_0 = G_1(0) \otimes \cdots \otimes G_d(0) =: \mathbf{I}.$$

Note that $G_i(0)$ is simply the identity matrix in $\mathbb{R}^{n_i \times n_i}$. With the aid of this definition, we now reformulate (5.2.4) in a more compact way as

$$\frac{\partial}{\partial t}\mathbf{P}(t) = \left(\sum_{\mu=1}^{\mathcal{M}} (\mathbf{G}_\mu - \mathbf{I}) \cdot \operatorname{diag}(\mathbf{a}_\mu) \right) \cdot \mathbf{P}(t), \tag{5.2.7}$$

where we define $\operatorname{diag}(\mathbf{a}_\mu)$ to be the tensor product of matrices containing the entries of $\left(\mathbf{a}_\mu^{(1)}\right)_{k,:}, \ldots, \left(\mathbf{a}_\mu^{(d)}\right)_{k,:}$ as diagonals, i.e.

$$\operatorname{diag}(\mathbf{a}_\mu) = \sum_{k=1}^{r_\mu} \operatorname{diag}\left(\left(\mathbf{a}_\mu^{(1)}\right)_{k,:}\right) \otimes \cdots \otimes \operatorname{diag}\left(\left(\mathbf{a}_\mu^{(d)}\right)_{k,:}\right), \tag{5.2.8}$$

for $\mu = 1, \ldots, \mathcal{M}$.

Theorem 5.2.2. *For any state $X = (x_1, \ldots, x_d)^T \in \mathcal{S}$, it holds that*

$$\left(\frac{\partial}{\partial t}\mathbf{P}(t)\right)_{x_1,\ldots,x_d} = \frac{\partial}{\partial t}P(X,t).$$

The proof of Theorem 5.2.2 can be found in Appendix A.1.2 as well as in [6]. In what follows, we will refer to the tensor operator

$$\mathbf{A} = \sum_{\mu=1}^{\mathcal{M}} (\mathbf{G}_\mu - \mathbf{I}) \cdot \operatorname{diag}(\mathbf{a}_\mu) \tag{5.2.9}$$

as the master-equation operator. The MME (5.2.7) can then be written as $\frac{\partial}{\partial t}\mathbf{P}(t) = \mathbf{A} \cdot \mathbf{P}(t)$. The aim is to solve the tensor-based MME by using numerical integration schemes in order to compute stationary distributions as well as to analyze the transient behavior of the probability distributions. The resulting systems of linear equations can be solved by applying (M)ALS, see Section 4.

6

Nearest-Neighbor Interaction Systems in the Tensor-Train Format

An important question when modeling tensor-based networks is the ordering of the TT cores corresponding to different dimensions of the state space. Dimensions that strongly correlate should ideally be represented by adjacent TT cores. A specific type of networks are nearest-neighbor interaction systems, which can be expressed by a specific TT representation that we presented in [9]. We will see that, using the proposed TT decomposition, such a system corresponds to the topology of the TT format.

6.1. Nearest-Neighbor Interaction Systems

Consistent with the terminology of coupled cell systems, see e.g. [83], a *nearest-neighbor interaction system* (NNIS) is a network of interacting systems. These systems, called *cells*, are coupled in a chain or a ring, i.e. we consider a finite number of cells $\Theta_1, \ldots, \Theta_d$ only allowing single-cell events on Θ_i, $i \in \{1, \ldots, d\}$, and interactions involving two adjacent cells Θ_i and Θ_{i+1}, $i \in \{1, \ldots, d-1\}$. If the considered system is *cyclic*, we also include possible interactions between Θ_d and Θ_1. Figure 6.1 shows the possible coupling structures of an NNIS.

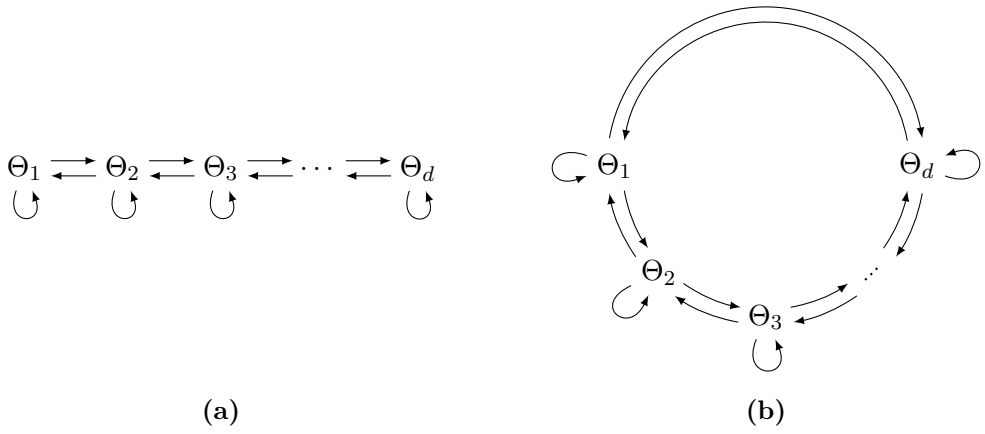


Figure 6.1: *Visualization of nearest-neighbor interaction systems: Events/interactions involve only one cell or two cells, respectively. (a) Visualization of a non-cyclic NNIS. (b) Visualization of a cyclic NNIS.*

Assuming that each cell Θ_i can be in n_i different states, which are identified by the set of natural numbers $\{1, \dots, n_i\}$, the state space $\mathcal{S}$ is given by

$$\mathcal{S} = \{1, \dots, n_1\} \times \{1, \dots, n_2\} \times \cdots \times \{1, \dots, n_d\}.$$

Thus, a state of the system is described by a vector $X = (x_1, \dots, x_d)^T \in \mathcal{S}$. In general, a tensor $\mathbf{T} \in \mathbb{R}^N = \mathbb{R}^{n_1 \times \cdots \times n_d}$ based on nearest-neighbor interactions can be expressed in the canonical format as

$$\mathbf{T}_{x_1,\dots,x_d} = \sum_{i=1}^{d} (\mathbf{S}_i)_{x_i} + \sum_{i=1}^{d-1} (\mathbf{K}_{i,i+1})_{x_i,x_{i+1}} + (\mathbf{K}_{d,1})_{x_d,x_1}\,, \tag{6.1.1}$$

with vectors $\mathbf{S}_i \in \mathbb{R}^{n_i}$ representing the single-cell events and matrices $\mathbf{K}_{i,i+1} \in \mathbb{R}^{n_i \times n_{i+1}}$ representing interactions between the cells Θ_i and Θ_{i+1}. The last term in (6.1.1) is only required for cyclic NNISs, i.e. the matrix $\mathbf{K}_{d,1}$ is only nonzero if there is at least one interaction between Θ_d and Θ_1. An NNIS is called *homogeneous* if the cell types and the events/interactions do not depend on the cell number, i.e.

$$\mathbf{S}_1 = \mathbf{S}_2 = \cdots = \mathbf{S}_d \quad \text{and} \quad \mathbf{K}_{1,2} = \mathbf{K}_{2,3} = \cdots = \mathbf{K}_{d-1,d}\,(= \mathbf{K}_{d,1}),$$

otherwise we call the system *heterogeneous*. Consequently, it also holds that the modes $n_1, \dots, n_d$ are equal for homogeneous systems. By computing QR decompositions of the matrices $K_{i,i+1}$, $i = 1, \dots, d$, we can construct canonical representations, i.e.

$$\mathbf{K}_{i,i+1} = Q \cdot R = \sum_{\mu=1}^{\beta_i} L_{i,\mu} \otimes M_{i+1,\mu}, \tag{6.1.2}$$

with $L_{i,\mu} \in \mathbb{R}^{n_i}$ being the μth column of Q, $M_{i+1,\mu} \in \mathbb{R}^{n_{i+1}}$ being the (transposed) μth row of R, and β_i being the matrix rank of $\mathbf{K}_{i,i+1}$ for $i = 1, \dots, d$. Note that we set $\mathbf{K}_{d,d+1} = \mathbf{K}_{d,1}$ and $M_{d+1,\mu} = M_{1,\mu}$. We can now reformulate (6.1.1) as

$$\mathbf{T}_{x_1,\dots,x_d} = \sum_{i=1}^{d} (\mathbf{S}_i)_{x_i} + \sum_{i=1}^{d} \sum_{\mu=1}^{\beta_i} (L_{i,\mu} \otimes M_{i+1,\mu})_{x_i,x_{i+1}}\,, \tag{6.1.3}$$

with $x_{d+1} = x_1$. Analogously to (6.1.1), a linear operator $\mathbf{A} \in \mathbb{R}^{N \times N}$ corresponding

to an NNIS can be expressed elementwise as

$$\mathbf{A}_{x_1,y_1,\ldots,x_d,y_d} = \sum_{i=1}^{d} (\mathbf{S}_i)_{x_i,y_i} + \sum_{i=1}^{d-1} (\mathbf{K}_{i,i+1})_{x_i,y_i,x_{i+1},y_{i+1}} + (\mathbf{K}_{d,1})_{x_d,y_d,x_1,y_1} \,. \quad (6.1.4)$$

Here, the components $\mathbf{S}_i$ are matrices and $\mathbf{K}_{i,i+1}$ are tensors of order 4. Similar to (6.1.3), by applying QR factorizations to the matricizations $\mathbf{K}_{i,i+1}\Big|_{n_i,n_i}^{n_{i+1},n_{i+1}}$, we obtain

$$\mathbf{A}_{x_1,y_1,\ldots,x_d,y_d} = \sum_{i=1}^{d} (\mathbf{S}_i)_{x_i,y_i} + \sum_{i=1}^{d} \sum_{\mu=1}^{\beta_i} (L_{i,\mu} \otimes M_{i+1,\mu})_{x_i,y_i,x_{i+1},y_{i+1}} \,, \quad (6.1.5)$$

with matrices $L_{i,\mu} \in \mathbb{R}^{n_i \times n_i}$ and $M_{i+1,\mu} \in \mathbb{R}^{n_{i+1} \times n_{i+1}}$ (with $i+1 \mathrel{\widehat{=}} 1$). That is, a tensor $\mathbf{T} \in \mathbb{R}^N$ and a tensor operator $\mathbf{A} \in \mathbb{R}^{N \times N}$, both describing an NNIS, have the same type of elementwise representation. Only the types of the components differ, i.e. the representations are either given by tensor products of vectors or matrices. As already considered in [5, 9], simple examples for tensors of this form are Ising models [84, 85] and linearly coupled oscillators [86, 87].

6.2. General SLIM Decomposition

In general, an NNIS can be represented by a canonical tensor constructed only with elementary tensors where at most two (adjacent) components are unequal to a vector of ones or to the identity matrix, respectively. That is, the tensor $\mathbf{T}$ given in (6.1.3) can be written as

$$\begin{aligned} \mathbf{T} \;=\; & \mathbf{S}_1 \otimes \mathbb{1}_2 \otimes \cdots \otimes \mathbb{1}_d \;+\; \ldots \;+\; \mathbb{1}_1 \otimes \cdots \otimes \mathbb{1}_{d-1} \otimes \mathbf{S}_d \\ & + \sum_{\mu=1}^{\beta_1} L_{1,\mu} \otimes M_{2,\mu} \otimes \mathbb{1}_3 \otimes \cdots \otimes \mathbb{1}_d \\ & + \ldots \\ & + \sum_{\mu=1}^{\beta_{d-1}} \mathbb{1}_1 \otimes \cdots \otimes \mathbb{1}_{d-2} \otimes L_{d-1,\mu} \otimes M_{d,\mu} \\ & + \sum_{\mu=1}^{\beta_d} M_{1,\mu} \otimes \mathbb{1}_2 \otimes \cdots \otimes \mathbb{1}_{d-1} \otimes L_{d,\mu}, \end{aligned} \quad (6.2.1)$$

with $\mathbb{1}_i = (1, \dots, 1)^T \in \mathbb{R}^{n_i}$ and the same components $\mathbf{S}_i$, $L_{i,\mu}$, and $M_{i+1,\mu}$ as in (6.1.3). Considering the tensor operator $\mathbf{A}$ given in (6.1.5), we can write

$$\begin{aligned}\mathbf{A} =& \mathbf{S}_1 \otimes I_2 \otimes \cdots \otimes I_d + \ldots + I_1 \otimes \cdots \otimes I_{d-1} \otimes \mathbf{S}_d \\ &+ \sum_{\mu=1}^{\beta_1} L_{1,\mu} \otimes M_{2,\mu} \otimes I_3 \otimes \cdots \otimes I_d \\ &+ \ldots \\ &+ \sum_{\mu=1}^{\beta_{d-1}} I_1 \otimes \cdots \otimes I_{d-2} \otimes L_{d-1,\mu} \otimes M_{d,\mu} \\ &+ \sum_{\mu=1}^{\beta_d} M_{1,\mu} \otimes I_2 \otimes \cdots \otimes I_{d-1} \otimes L_{d,\mu},\end{aligned} \tag{6.2.2}$$

with identity matrices $I_i \in \mathbb{R}^{n_i \times n_i}$ and the same components $\mathbf{S}_i$, $L_{i,\mu}$, and $M_{i+1,\mu}$ as in (6.1.5).

In what follows, we will only describe how to derive operator representations in the TT format since the derivations for (6.2.1) and (6.2.2) are almost identical. Considering a heterogeneous and cyclic NNIS, we first define the TT cores $\mathbf{L}_i$ and $\mathbf{M}_{i+1}$, $i = 1, \dots, d-1$, which contain all matrices $L_{i,\mu}$ and $M_{i+1,\mu}$, respectively, for $\mu = 1, \dots, \beta_i$, i.e.

$$\begin{aligned}[\mathbf{L}_i] &= \begin{bmatrix} L_{i,1} & \ldots & L_{i,\beta_i} \end{bmatrix} && \in \mathbb{R}^{1 \times n_i \times n_i \times \beta_i}, \\ [\mathbf{M}_{i+1}] &= \begin{bmatrix} M_{i+1,1} & \ldots & M_{i+1,\beta_i} \end{bmatrix}^{\mathbb{T}} && \in \mathbb{R}^{\beta_i \times n_{i+1} \times n_{i+1} \times 1}.\end{aligned}$$

Furthermore, for the interactions between the cells Θ_d and Θ_1, we define

$$\begin{aligned}[\mathbf{L}_d] &= \begin{bmatrix} L_{d,1} & \ldots & L_{d,\beta_d} \end{bmatrix}^{\mathbb{T}} \in \mathbb{R}^{\beta_d \times n_d \times n_d \times 1}, \\ [\mathbf{M}_1] &= \begin{bmatrix} M_{1,1} & \ldots & M_{1,\beta_d} \end{bmatrix} \in \mathbb{R}^{1 \times n_1 \times n_1 \times \beta_d}.\end{aligned}$$

Here, we use the core notation and the definition of a rank-transposed TT core, see Section 3.4.1. As we already discussed in Section 3.4.5, it is often possible to derive more compact tensor decompositions in the TT format than in the canonical format. Let $\mathbf{I}_i = I_i \in \mathbb{R}^{n_i \times n_i}$ and $\mathbf{J}_i \in \mathbb{R}^{\beta_d \times n_i \times n_i \times \beta_d}$ be a TT core with

$$[\mathbf{J}_i] = \begin{bmatrix} I_i & & 0 \\ & \ddots & \\ 0 & & I_i \end{bmatrix}.$$

Since the rank-one tensors of the canonical representation (6.2.2) differ only in a small number of cores, we can write $\mathbf{A}$ as a TT decomposition given by

$$\begin{aligned}\mathbf{A} = \begin{bmatrix}\mathbf{S}_1 & \mathbf{L}_1 & \mathbf{I}_1 & \mathbf{M}_1\end{bmatrix} \otimes \begin{bmatrix}\mathbf{I}_2 & 0 & 0 & 0\\ \mathbf{M}_2 & 0 & 0 & 0\\ \mathbf{S}_2 & \mathbf{L}_2 & \mathbf{I}_2 & 0\\ 0 & 0 & 0 & \mathbf{J}_2\end{bmatrix} \otimes \cdots \\ \cdots \otimes \begin{bmatrix}\mathbf{I}_{d-1} & 0 & 0 & 0\\ \mathbf{M}_{d-1} & 0 & 0 & 0\\ \mathbf{S}_{d-1} & \mathbf{L}_{d-1} & \mathbf{I}_{d-1} & 0\\ 0 & 0 & 0 & \mathbf{J}_{d-1}\end{bmatrix} \otimes \begin{bmatrix}\mathbf{I}_d\\ \mathbf{M}_d\\ \mathbf{S}_d\\ \mathbf{L}_d\end{bmatrix}.\end{aligned} \tag{6.2.3}$$

From now on, we will call the TT decomposition given in (6.2.3) *SLIM decomposition*. The origin of this term is explained by the structure of the first core. The proof of the next theorem can be found in Appendix A.1.3.

Theorem 6.2.1. *The SLIM decomposition given in* (6.2.3) *corresponds to the canonical decomposition given in* (6.2.2).

The TT ranks of the decomposition (6.2.3) are given by $r_0 = r_d = 1$ and $r_i = 2 + \beta_i + \beta_d$ for $i = 1, \dots, d-1$. To reduce the storage consumption, the different TT cores can be stored as sparse arrays.

Lemma 6.2.2. *The storage consumption of the SLIM decomposition* (6.2.3) *(in sparse format) can be estimated as*

$$O\left(\sum_{i=1}^{d}(\beta_{i-1}+\beta_i+1)\cdot n_i^2 + \sum_{i=2}^{d-1}(\beta_d+2)\cdot n_i + n_1 + n_d\right),$$

with $\beta_0 = \beta_d$.

Proof. We assume the matrices of the core elements $\mathbf{S}_i$, $\mathbf{L}_i$, and $\mathbf{M}_i$, $i = 1, \dots, d$, to be dense, i.e. the storage consumption of a single matrix is then estimated as $O(n_i^2)$. Since $\mathbf{I}_i = I \in \mathbb{R}^{n_i \times n_i}$ has only n_i entries, we obtain $O(n_i)$ for the components $\mathbf{I}_i$. Analogously, we can estimate the storage of $\mathbf{J}_i$ as $O(\beta_d \cdot n_i)$. Thus, we obtain the following storage estimates for the different TT cores:

$$\begin{aligned}\mathbf{A}^{(1)} &: O\left((\beta_d+\beta_1+1)n_1^2 + n_1\right),\\ \mathbf{A}^{(i)}, 2 \le i \le d-1 &: O\left((\beta_{i-1}+\beta_i+1)n_i^2 + (2+\beta_d)n_i\right),\\ \mathbf{A}^{(d)} &: O\left((\beta_{d-1}+\beta_d+1)n_d^2 + n_d\right).\end{aligned}$$

Summation over all cores concludes the proof. □

Assuming that the ranks β_i and dimensions n_i are bounded (or even fixed) for an increasing number of cells, we obtain a linear growth of the storage consumption. The SLIM decomposition (6.2.3) holds for all heterogeneous and cyclic NNISs. For homogeneous systems, it can be simplified to

$$\mathbf{A} = \begin{bmatrix} \mathbf{S} & \mathbf{L} & \mathbf{I} & \mathbf{M} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I} & 0 & 0 & 0 \\ \mathbf{M} & 0 & 0 & 0 \\ \mathbf{S} & \mathbf{L} & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{J} \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \mathbf{I} & 0 & 0 & 0 \\ \mathbf{M} & 0 & 0 & 0 \\ \mathbf{S} & \mathbf{L} & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{J} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I} \\ \mathbf{M} \\ \mathbf{S} \\ \mathbf{L} \end{bmatrix}. \quad (6.2.4)$$

Thus, the advantage of the SLIM decomposition is the repeating pattern of TT cores. That is, if we increase or decrease the number of cells, we only insert or remove a TT core, respectively. Since a cyclic system has at least three cells (two coupled cells are represented by (6.2.5)), the first and the last core in (6.2.4) remain fixed while the number of cores in between is arbitrary, but must be greater than zero. If there are no interactions between the cells Θ_d and Θ_1, the SLIM decomposition for a heterogeneous NNIS is given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{S}_1 & \mathbf{L}_1 & \mathbf{I}_1 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I}_2 & 0 & 0 \\ \mathbf{M}_2 & 0 & 0 \\ \mathbf{S}_2 & \mathbf{L}_2 & \mathbf{I}_2 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I}_{d-1} & 0 & 0 \\ \mathbf{M}_{d-1} & 0 & 0 \\ \mathbf{S}_{d-1} & \mathbf{L}_{d-1} & \mathbf{I}_{d-1} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I}_d \\ \mathbf{M}_d \\ \mathbf{S}_d \end{bmatrix}. \quad (6.2.5)$$

If the NNIS is additionally homogeneous, we obtain the simplest form of a SLIM decomposition:

$$\mathbf{A} = \begin{bmatrix} \mathbf{S} & \mathbf{L} & \mathbf{I} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I} & 0 & 0 \\ \mathbf{M} & 0 & 0 \\ \mathbf{S} & \mathbf{L} & \mathbf{I} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I} & 0 & 0 \\ \mathbf{M} & 0 & 0 \\ \mathbf{S} & \mathbf{L} & \mathbf{I} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I} \\ \mathbf{M} \\ \mathbf{S} \end{bmatrix}.$$

6.3. SLIM Decomposition for Markov Generators

In our work, we are particularly interested in NNISs corresponding to Markovian master equations, cf. Section 5. The interactions – denoted as elementary reactions R – in such systems can be described in one of the following forms:

$$\begin{array}{lllll} \text{(i)} & \mathrm{R} : x_i & \to & y_i, & \\ \text{(ii)} & \mathrm{R} : x_i, x_{i+1} & \to & y_i, y_{i+1}, & \\ \text{(iii)} & \mathrm{R} : x_d, x_1 & \to & y_d, y_1. & \end{array} \quad (6.3.1)$$

For $i = 1, \ldots, d$, the values $x_i, y_i \in \{1, \ldots, n_i\}$ represent the state of Θ_i before and after the event R was executed, respectively.

A *single-cell reaction* (SCR) only changes the state of one cell, whereas a *two-cell reaction* (TCR) changes the states of two adjacent cells. TCRs of the form (iii) only occur in cyclic systems. Furthermore, elementary reactions either depend only on the state of a single cell or on the states of two adjacent cells. That is, any reaction propensity, cf. (5.2.6), corresponding to an SCR $\mathrm{R}_{i,\nu}$ on cell Θ_i has the form

$$\mathbf{a}_{i,\nu} = \mathbb{1}_1 \otimes \cdots \otimes \mathbb{1}_{i-1} \otimes \mathrm{a}_{i,\nu} \otimes \mathbb{1}_{i+1} \otimes \cdots \otimes \mathbb{1}_d, \tag{6.3.2}$$

with $a_{i,\nu} \in \mathbb{R}^{n_i}$ and $\nu = 1, \ldots, \alpha_i$, where $\alpha_i \in \mathbb{N}$ is the number of all SCRs on Θ_i. The reaction propensity $\mathbf{a}_{i,i+1,\mu}$ corresponding to a TCR $\mathrm{R}_{i,i+1,\mu}$ acting on the cells Θ_i and Θ_{i+1}, $i = 1, \ldots, d-1$, can be expressed as

$$\mathbf{a}_{i,i+1,\mu} = \mathbb{1}_1 \otimes \cdots \otimes \mathbb{1}_{i-1} \otimes \mathrm{a}_{i,i+1,\mu} \otimes \mathbb{1}_{i+2} \otimes \cdots \otimes \mathbb{1}_d, \tag{6.3.3}$$

with $\mathrm{a}_{i,i+1,\mu} \in \mathbb{R}^{n_i \times n_{i+1}}$ and $\mu = 1, \ldots, \beta_i$, where $\beta_i \in \mathbb{N}$ is the number of all TCRs between Θ_i and Θ_{i+1}. Similar to (6.1.2), we decompose $\mathrm{a}_{i,i+1,\mu}$ into

$$\mathrm{a}_{i,i+1,\mu} = \sum_{k=1}^{r_{i,i+1,\mu}} \left(\mathrm{a}^{(1)}_{i,i+1,\mu}\right)_{k,:} \otimes \left(\mathrm{a}^{(2)}_{i,i+1,\mu}\right)_{k,:}.$$

Thus, the reaction propensities can be written as

$$\mathbf{a}_{i,i+1,\mu} = \sum_{k=1}^{r_{i,i+1,\mu}} \mathbb{1}_2 \otimes \cdots \otimes \mathbb{1}_{i-1} \otimes \left(\mathrm{a}^{(1)}_{i,i+1,\mu}\right)_{k,:} \otimes \left(\mathrm{a}^{(2)}_{i,i+1,\mu}\right)_{k,:} \otimes \mathbb{1}_{i+2} \otimes \cdots \otimes \mathbb{1}_d.$$

For the propensity tensor corresponding to the reactions between Θ_d and Θ_1, we obtain

$$\mathbf{a}_{d,1,\mu} = \sum_{k=1}^{r_{d,1,\mu}} \left(\mathrm{a}^{(2)}_{d,1,\mu}\right)_{k,:} \otimes \mathbb{1}_2 \otimes \cdots \otimes \mathbb{1}_{d-1} \otimes \left(\mathrm{a}^{(1)}_{d,1,\mu}\right)_{k,:}, \tag{6.3.4}$$

for $\mu = 1, \ldots, \beta_d$. The representation (6.3.4) can be derived by decomposing a permuted propensity tensor $\tilde{\mathbf{a}}_{d,1,\mu}$, cf. (3.5.4) and (3.5.5), with

$$\left(\tilde{\mathbf{a}}_{d,1,\mu}\right)_{x_2,\ldots,x_d,x_1} = \left(\mathbf{a}_{d,1,\mu}\right)_{x_1,x_2,\ldots,x_d},$$

and rotating the cores back, see [9] for further details. For the reactions (6.3.1), we

then obtain the following diagonalizations of the propensity tensors:

(i) $I \otimes \cdots \otimes I \otimes \operatorname{diag}(\mathrm{a}_{i,\nu}) \otimes I \otimes \cdots \otimes I$,

(ii) $\sum_{k=1}^{r_{i,i+1,\mu}} I \otimes \cdots \otimes I \otimes \operatorname{diag}\left(\left(\mathrm{a}^{(1)}_{i,i+1,\mu}\right)_{k,:}\right) \otimes \operatorname{diag}\left(\left(\mathrm{a}^{(2)}_{i,i+1,\mu}\right)_{k,:}\right) \otimes I \otimes \cdots \otimes I$,

(iii) $\sum_{k=1}^{r_{d,1,\mu}} \operatorname{diag}\left(\left(\mathrm{a}^{(2)}_{d,1,\mu}\right)_{k,:}\right) \otimes I \otimes \cdots \otimes I \otimes \operatorname{diag}\left(\left(\mathrm{a}^{(1)}_{d,1,\mu}\right)_{k,:}\right)$,

for $\nu = 1, \ldots, \alpha_i$ and $\mu = 1, \ldots, \beta_i$. For an SCR $\mathrm{R}_{i,\nu}$, the vectors of net changes, cf. (5.2.2), have the form

$$\xi_{i,\nu} = (0, \ldots, 0, p_{i,\nu}, 0, \ldots, 0)^T, \tag{6.3.5}$$

with $p_{i,\nu} \in \mathbb{Z}$. Considering a TCR $\mathrm{R}_{i,i+1,\mu}$, it holds that

$$\xi_{i,i+1,\mu} = (0, \ldots, 0, p_{i,i+1,\mu}, q_{i,i+1,\mu}, 0, \ldots, 0)^T, \tag{6.3.6}$$

with $p_{i,i+1,\mu}, q_{i,i+1,\mu} \in \mathbb{Z}$. Following Definition 5.2.1, the multidimensional shift operators corresponding to the SCRs and TCRs, respectively, are given by

(i) $\mathbf{G}_{i,\nu} = I \otimes \cdots \otimes I \otimes G_i(-p_{i,\nu}) \otimes I \otimes \cdots \otimes I$,

(ii) $\mathbf{G}_{i,i+1,\mu} = I \otimes \cdots \otimes I \otimes G_i(-p_{i,i+1,\mu}) \otimes G_{i+1}(-q_{i,i+1,\mu}) \otimes I \otimes \cdots \otimes I$,

(iii) $\mathbf{G}_{d,1,\mu} = G_1(-q_{d,1,\mu}) \otimes I \otimes \cdots \otimes I \otimes G_d(-p_{d,1,\mu})$.

Considering the MME (5.2.7), we can now write the master-equation operator $\mathbf{A}$ defined in (5.2.9) as

$$\mathbf{A} = \sum_{i=1}^{d} \sum_{\nu=1}^{\alpha_i} \mathbf{A}_{i,\nu} + \sum_{i=1}^{d-1} \sum_{\mu=1}^{\beta_i} \mathbf{A}_{i,i+1,\mu} + \sum_{\mu=1}^{\beta_d} \mathbf{A}_{d,1,\mu}, \tag{6.3.7}$$

with

(i) $\mathbf{A}_{i,\nu} = (\mathbf{G}_{i,\nu} - \mathbf{I}) \cdot \operatorname{diag}(\mathbf{a}_{i,\mu})$,

(ii) $\mathbf{A}_{i,i+1,\mu} = (\mathbf{G}_{i,i+1,\mu} - \mathbf{I}) \cdot \operatorname{diag}(\mathbf{a}_{i,i+1,\mu})$,

(iii) $\mathbf{A}_{d,1,\mu} = (\mathbf{G}_{d,1,\mu} - \mathbf{I}) \cdot \operatorname{diag}(\mathbf{a}_{d,1,\mu})$.

Identifying the cell pair (Θ_d, Θ_1) and all corresponding quantities with (Θ_d, Θ_{d+1}), we now define the matrices

$$\begin{aligned}
\tilde{S}_{i,\nu} &= \operatorname{diag}\left(\mathrm{a}_{i,\nu}\right), & S_{i,\nu} &= G_i(-p_{i,\nu}) \cdot \tilde{S}_{i,\nu}, \\
\tilde{L}_{i,\mu,k} &= \operatorname{diag}\left(\left(\mathrm{a}^{(1)}_{i,i+1,\mu}\right)_{k,:}\right), & L_{i,\mu,k} &= G_i(-p_{i,i+1,\mu}) \cdot \tilde{L}_{i,\mu,k}, \\
\tilde{M}_{i+1,\mu,k} &= \operatorname{diag}\left(\left(\mathrm{a}^{(2)}_{i,i+1,\mu}\right)_{k,:}\right), & M_{i+1,\mu,k} &= G_{i+1}(-q_{i,i+1,\mu}) \cdot \tilde{M}_{i+1,\mu,k},
\end{aligned} \tag{6.3.8}$$

for $i = 1, \ldots, d$, $\nu = 1, \ldots, \alpha_i$, $\mu = 1, \ldots, \beta_i$, and $k = 1, \ldots, r_{i,i+1,\mu}$. Due to the bilinearity of the tensor product, see Section 2.2.4, we define

$$\mathbf{S}_i = \sum_{\nu=1}^{\alpha_i} \left(S_{i,\nu} - \tilde{S}_{i,\nu} \right). \tag{6.3.9}$$

Moreover, we gather all the matrices $L_{i,\mu,k}, \tilde{L}_{i,\mu,k}$ and $M_{i+1,\mu,k}, \tilde{M}_{i+1,\mu,k}$ in the TT cores $\mathbf{L}_i$ and $\mathbf{M}_{i+1}$, respectively. The cores are then defined as

$$\begin{aligned}
[\mathbf{L}_i] &= \underbrace{\begin{bmatrix} L_{i,1,1} & -\tilde{L}_{i,1,1} & \ldots & L_{i,\beta_i,r_{i,i+1,\beta_i}} & -\tilde{L}_{i,\beta_i,r_{i,i+1,\beta_i}} \end{bmatrix}}_{\in \mathbb{R}^{1 \times n_i \times n_i \times (\beta_i \cdot r_{i,i+1,\beta_i})}}, \\
[\mathbf{M}_{i+1}] &= \underbrace{\begin{bmatrix} M_{i+1,1,1} & \tilde{M}_{i+1,1,1} & \ldots & M_{i+1,\beta_i,r_{i,i+1,\beta_i}} & \tilde{M}_{i+1,\beta_i,r_{i,i+1,\beta_i}} \end{bmatrix}^{\mathbb{T}}}_{\in \mathbb{R}^{(\beta_i \cdot r_{i,i+1,\beta_i}) \times n_{i+1} \times n_{i+1} \times 1}},
\end{aligned} \tag{6.3.10}$$

for $i = 1, \ldots, d-1$, and

$$\begin{aligned}
[\mathbf{L}_d] &= \underbrace{\begin{bmatrix} L_{d,1,1} & -\tilde{L}_{d,1,1} & \ldots & L_{d,\beta_d,r_{d,1,\beta_d}} & -\tilde{L}_{d,\beta_d,r_{d,1,\beta_d}} \end{bmatrix}^{\mathbb{T}}}_{\in \mathbb{R}^{(\beta_d \cdot r_{d,1,\beta_d}) \times n_d \times n_d \times 1}}, \\
[\mathbf{M}_1] &= \underbrace{\begin{bmatrix} M_{1,1,1} & \tilde{M}_{1,1,1} & \ldots & M_{1,\beta_d,r_{d,1,\beta_d}} & \tilde{M}_{1,\beta_d,r_{d,1,\beta_d}} \end{bmatrix}}_{\in \mathbb{R}^{1 \times n_1 \times n_1 \times (\beta_d \cdot r_{d,1,\beta_d})}},
\end{aligned} \tag{6.3.11}$$

for the reactions on the cell pair (Θ_d, Θ_1), respectively. Here, we use the notation for rank-transposed cores given in (3.4.4). The TT cores above can now be inserted into (6.2.3) resulting in the SLIM decomposition of the generator $\mathbf{A}$.

Theorem 6.3.1. *The SLIM decomposition of the form* (6.2.3) *with components given in* (6.3.9), (6.3.10), *and* (6.3.11) *is a TT representation of the MME operator* (6.3.7).

The proof of Theorem 6.3.1 can be found in Appendix A.1.4. Note that the application of Algorithm 15, see Appendix A.2.6, may reduce the number of components within the above TT cores, since (6.3.10) and (6.3.11) do not necessarily have to be the smallest possible cores (in terms of TT ranks) in order to represent the core products $[\mathbf{L}_i] \otimes [\mathbf{M}_{i+1}]$ and $[\mathbf{M}_1] \otimes [\mathbf{L}_d]$, respectively. These core products are essential parts of SLIM decompositions, see the proof of Theorem 6.2.1 in Appendix A.1.3. In lines 3 and 4 of Algorithm 15 we use the multi-index notation as defined in (2.4.2). The automatic construction of the SLIM decomposition of a master-equation operator corresponding to a (cyclic or non-cyclic) NNIS is implemented in Algorithm 16, see Appendix A.2.7. If the NNIS is cyclic, we again set $\Theta_{d+1} = \Theta_1$, $n_{d+1} = n_1$, $\mathrm{R}_{d,d+1,\mu} = \mathrm{R}_{d,1,\mu}$ and so forth.

In Chapters 9 and 10, we will show examples for SLIM decompositions on high-dimensional state spaces, where the corresponding master-equation operators describe chemical reaction networks and heterogeneous catalytic processes, respectively. Additionally, we gave an example for a traffic problem in [9], where we computed the distribution of cars at a toll station. Since many different physical and biological systems can be represented as NNISs, SLIM decompositions may also be exploited in further application areas.

7

Dynamic Mode Decomposition in the Tensor-Train Format

In this chapter, we will derive a tensor-based version of the dynamic mode decomposition [56, 57]. For this purpose, we will describe how to compute pseudoinverses of certain matricizations of a tensor $\mathbf{T} \in \mathbb{R}^N$. As we have shown in [14], information about these pseudoinverses can be directly deduced from the TT representation of $\mathbf{T}$ after some orthonormalization steps. Using this information, we are able to compute pseudoinverses of tensor unfoldings without necessitating the solution of an optimization problem, cf. [88], and therefore can construct exact TT decompositions of DMD modes.

7.1. Moore-Penrose Inverse

In classical linear algebra, the *pseudoinverse* (or *Moore–Penrose inverse*) of a matrix $A \in \mathbb{R}^{m \times n}$ is a generalization of the inverse matrix.

Definition 7.1.1. *Given a matrix $A \in \mathbb{R}^{m \times n}$, the pseudoinverse $A^+ \in \mathbb{R}^{n \times m}$ is defined by the properties*

$$\begin{aligned} &\text{(i)} && AA^+A = A, \\ &\text{(ii)} && A^+AA^+ = A^+, \\ &\text{(iii)} && (AA^+)^T = AA^+, \\ &\text{(iv)} && (A^+A)^T = A^+A. \end{aligned}$$

The pseudoinverse A^+ is unique for all real or complex matrices A. Given an (underdetermined or overdetermined) system of linear equations of the form $Ax = y$ with $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$, the vector A^+y is a solution of the least-squares problem

$$\min_{x \in \mathbb{R}^n} \|Ax - y\|_2 . \tag{7.1.1}$$

Furthermore, it can be computed by a (compact/reduced) SVD. That is, we assume that the SVD of A is given by

$$A = U \, \Sigma \, V^T, \tag{7.1.2}$$

where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ with $U^TU = V^TV = I$. As in the previous chapters, $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r\times r}$ denotes the diagonal matrix containing only the non-zero singular values (sorted in decreasing order) of A. The pseudoinverse A^+ can then be computed by

$$A^+ = V\,\Sigma^{-1}\,U^T, \tag{7.1.3}$$

where $\Sigma^{-1} = \operatorname{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1})$.

7.2. Computation of the Pseudoinverse

Given a tensor $\mathbf{T} \in \mathbb{R}^N$ in TT format, i.e. $\mathbf{T} = \left[\mathbf{T}^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{T}^{(d)}\right]$, we consider a matricization (or unfolding)

$$T = \mathbf{T}\Big|_{n_1,\ldots,n_l}^{n_{l+1},\ldots,n_d},$$

where $1 \leq l \leq d$, see Section 2.4. In order to directly construct the pseudoinverse T^+ from a TT representation of $\mathbf{T}$, we apply the two orthonormalization procedures shown in Algorithms 4 and 5. The difference to Algorithm 9 and Algorithm 10, respectively, is that we here do not orthonormalize the complete tensor train. That is, Algorithm 4 left-orthonormalizes the TT cores from $\mathbf{T}^{(1)}$ to $\mathbf{T}^{(l)}$ for $1 \leq l \leq d-1$ while Algorithm 5 right-orthonormalizes the cores from $\mathbf{T}^{(l)}$ to $\mathbf{T}^{(d)}$ for $2 \leq l \leq d$.

Similar to the statements in Section 3.4.3, both algorithms compute different but equivalent representations of the tensor $\mathbf{T}$. By using SVDs instead of QR factorizations, it is also possible to truncate the TT cores during the orthonormalization processes, cf. Algorithm 2. However, this would then result in the pseudoinverse of an approximation of the given tensor $\mathbf{T}$.

Algorithm 4 Partial left-orthonormalization of tensor trains

Input: Tensor train $\mathbf{T} \in \mathbb{R}^N$ with TT cores $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{i-1}\times n_i\times r_i}$, $i = 1, \ldots, d$, and core number l, $1 \leq l \leq d-1$.

Output: Tensor train $\mathbf{T}$ with left-orthonormal cores $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(l)}$.

1: **for** $i = 1, \ldots, l$ **do**
2: Compute QR factorization of the left-unfolding $\mathcal{L}\left(\mathbf{T}^{(i)}\right)$, i.e. $\mathcal{L}\left(\mathbf{T}^{(i)}\right) = Q \cdot R$ with $Q \in \mathbb{R}^{r_{i-1}\cdot n_i\times s}$ and $Q^T \cdot Q = I$.
3: Define $\mathbf{U} \in \mathbb{R}^{r_{i-1}\times n_i\times s}$ as a reshaped version of Q with $\mathbf{U}_{k,x,l} = Q_{\overline{k,x},l}$.
4: Define $\mathbf{V} \in \mathbb{R}^{s\times n_{i+1}\times r_{i+1}}$ by $\mathcal{R}(\mathbf{V}) = R \cdot \mathcal{R}\left(\mathbf{T}^{(i+1)}\right)$.
5: Set $\mathbf{T}^{(i)}$ to $\mathbf{U}$, $\mathbf{T}^{(i+1)}$ to $\mathbf{V}$, and r_i to s.
6: **end for**

Algorithm 5 Partial right-orthonormalization of tensor trains

Input: Tensor train $\mathbf{T} \in \mathbb{R}^N$ with TT cores $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$, $i = 1, \dots, d$, and core number l, $2 \leq l \leq d$.

Output: Tensor train $\mathbf{T}$ with right-orthonormal cores $\mathbf{T}^{(l)}, \dots, \mathbf{T}^{(d)}$.

1: **for** $i = d, \dots, l$ **do**
2: Compute QR factorization of the right-unfolding $\left(\mathcal{R}\left(\mathbf{T}^{(i)}\right)\right)^T$, i.e. $\mathcal{R}\left(\mathbf{T}^{(i)}\right) = R^T \cdot Q^T$ with $Q^T \in \mathbb{R}^{s \times n_i \cdot r_i}$ and $Q^T \cdot Q = I$.
3: Define $\mathbf{U} \in \mathbb{R}^{s \times n_i \times r_i}$ as a reshaped version of Q^T with $\mathbf{U}_{k,x,l} = Q^T_{k,\overline{x,l}}$.
4: Define $\mathbf{V} \in \mathbb{R}^{r_{i-2} \times n_{i-1} \times s}$ by $\mathcal{L}(\mathbf{V}) = \mathcal{L}\left(\mathbf{T}^{(i-1)}\right) \cdot R^T$.
5: Set $\mathbf{T}^{(i)}$ to $\mathbf{U}$, $\mathbf{T}^{(i-1)}$ to $\mathbf{V}$, and r_{i-1} to s.
6: **end for**

With the aid of Algorithms 4 and 5, pseudoinverses of arbitrary tensor unfoldings T with respect to the dimensions $(1, \dots, l)$ and $(l+1, \dots, d)$ can be constructed by computing a global SVD of the whole tensor train. After left-orthonormalizing the cores $\mathbf{T}^{(1)}, \dots, \mathbf{T}^{(l)}$ and right-orthonormalizing the cores $\mathbf{T}^{(l+1)}, \dots, \mathbf{T}^{(d)}$, the pseudoinverse T^+ can be obtained by reordering the cores. This is illustrated in Figure 7.1. Algorithm 6 shows the procedure for computing the pseudoinverse. Note that the pseudoinverse depends on the matricization of the tensor $\mathbf{T}$. For different matricizations, we also obtain different pseudoinverses.

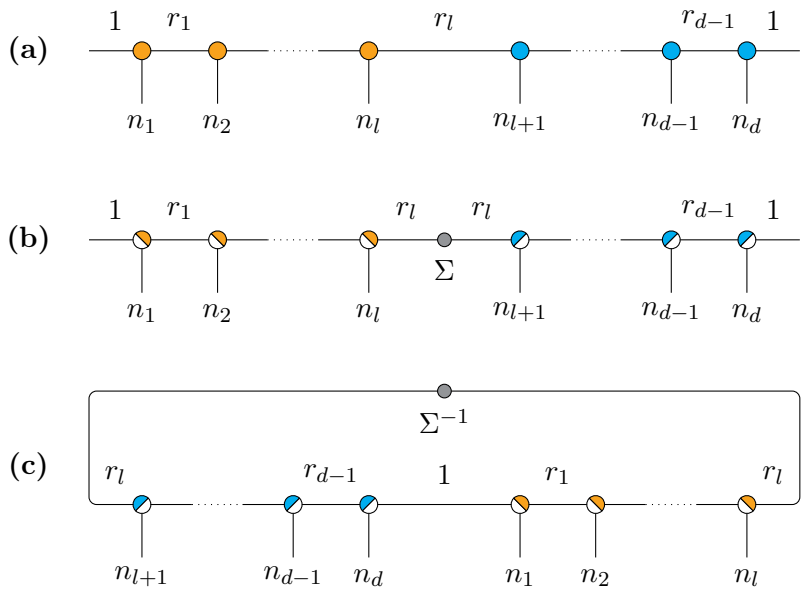


Figure 7.1: *Computation of the pseudoinverse of a tensor train: a) Initial tensor* $\mathbf{T}$*. b) Left- and right-orthonormalization of the tensor cores. c) Representation of the pseudoinverse* $\mathbf{T}^+$*.*

Algorithm 6 Pseudoinversion of tensor trains.

Input: Tensor train $\mathbf{T} \in \mathbb{R}^N$ and core number l, $2 \leq l \leq d$.

Output: Pseudoinverse of $T = \mathbf{T} \Big|_{n_1,\ldots,n_l}^{n_{l+1},\ldots,n_d}$.

1: Left-orthonormalize $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(l-1)}$ and right-orthonormalize $\mathbf{T}^{(l+1)}, \ldots, \mathbf{T}^{(d)}$ using Algorithms 4 and 5.
2: Compute SVD of $\mathcal{L}\left(\mathbf{T}^{(l)}\right)$, i.e. $\mathcal{L}\left(\mathbf{T}^{(l)}\right) = U\Sigma V^T$ with $\Sigma \in \mathbb{R}^{s\times s}$.
3: Define $\mathbf{U} \in \mathbb{R}^{r_{l-1}\times n_l \times s}$ as a reshaped version of U with $\mathbf{U}_{k,x,l} = U_{\overline{k,x},l}$.
4: Define $\mathbf{V} \in \mathbb{R}^{s\times n_{l+1}\times r_{l+1}}$ by $\mathcal{R}(\mathbf{V}) = V^T \cdot \mathcal{R}\left(\mathbf{T}^{(l+1)}\right)$.
5: Set $\mathbf{T}^{(l)}$ to $\mathbf{U}$, $\mathbf{T}^{(l+1)}$ to $\mathbf{V}$, and r_l to s.
6: Define $\tilde{U} = \left(\sum_{k_0=1}^{r_0} \cdots \sum_{k_{l-1}=1}^{r_{l-1}} \mathbf{T}^{(1)}_{k_0,:,k_1} \otimes \ldots \otimes \mathbf{T}^{(l)}_{k_{l-1},:,:}\right) \Big|_{n_1,\ldots,n_l}^{r_l}$.
7: Define $\tilde{V} = \left(\sum_{k_{l+1}=1}^{r_{l+1}} \cdots \sum_{k_d=1}^{r_d} \mathbf{T}^{(l+1)}_{:,:,k_{l+1}} \otimes \ldots \otimes \mathbf{T}^{(d)}_{k_{d-1},:,k_d}\right) \Big|_{n_{l+1},\ldots,n_d}^{r_l}$.
8: Define $T^+ = \tilde{V}\,\Sigma^{-1}\,\tilde{U}^T$.

Theorem 7.2.1. *Given a tensor* $\mathbf{T}$ *and core number* $1 \leq l \leq d-1$*, Algorithm 6 computes the pseudoinverse with respect to the dimensions* $(1,\ldots,l)$ *and* $(l+1,\ldots,d)$*.*

The proof of Theorem 7.2.1 can be found in Appendix A.1.5. The algorithm above computes the pseudoinverse of the matricization of $\mathbf{T}$ with respect to the dimensions $(1,\ldots,l)$ and $(l+1,\ldots,d)$ whether or not the TT cores of $\mathbf{T}$ are orthonormal. If, for instance, all TT cores $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(d-1)}$ are already left-orthonormal, the application of Algorithm 4 can be skipped. The algorithm is then modified such that only the cores $\mathbf{T}^{(l+2)}, \ldots, \mathbf{T}^{(d)}$ are right-orthonormalized.

An important aspect is that we do not need to compute the pseudoinverse of T explicitly. Instead, we only orthonormalize the TT cores and compute the matrix Σ by executing the lines 1 to 5 of Algorithm 6. We then store the representation

$$\mathbf{T}^+ = \sum_{k_1=1}^{r_1} \cdots \sum_{k_{d-1}=1}^{r_{d-1}} \sigma_{k_l}^{-1} \cdot \mathbf{T}^{(l+1)}_{k_l,:,k_{l+1}} \otimes \ldots \otimes \mathbf{T}^{(d)}_{k_{d-1},:,1} \otimes \mathbf{T}^{(1)}_{1,:,k_1} \otimes \ldots \otimes \mathbf{T}^{(l)}_{k_{l-1},:,k_l},$$

which can be either regarded as the sum of r_l tensor trains scaled by $\sigma_1^{-1}, \ldots, \sigma_{r_l}^{-1}$ or as a cyclic tensor train as depicted in Figure 7.1, cf. Section 3.5.3.

7.3. Tensor-Based Dynamic Mode Decomposition

First introduced in 2008 by Schmid et al. [56, 57], *dynamic mode decomposition* (DMD) is widely used to identify low-order dynamics by decomposing high-dimensional data into coupled spatial-temporal modes [58]. Applied to time-series flow field data, these modes often correspond to coherent structures in the flow. DMD constitutes an effective tool for the analysis of the behavior of complex dy-

namical systems and is related to the *principle component analysis* [89]. Based on our work in [14], we show an extension of DMD – so-called *tensor-based dynamic mode decomposition* (TDMD) – that exploits the TT format in order to compute DMD modes and corresponding eigenvalues. In this way, we may reduce the computational complexity and the storage consumption, which enables us to mitigate the curse of dimensionality when considering high-dimensional dynamical systems. In Chapter 11, we will give different examples from fluid dynamics to illustrate the efficiency of TDMD.

Consider a fluid flow on a domain $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$, e.g., described by Navier–Stokes equations, see Section 11.1. We assume that the flow field is interpolated on a rectangular grid at equidistant time points and represented by a set of m snapshots

$$\mathbf{T}_0, \mathbf{T}_1, \ldots, \mathbf{T}_m. \tag{7.3.1}$$

For instance, each tensor $\mathbf{T}_k$ could contain the vorticities or the velocity magnitudes at the grid points corresponding to the interpolated flow field data such that $\mathbf{T}_k \in \mathbb{R}^N = \mathbb{R}^{n_1 \times \cdots \times n_d}$ with the modes n_i being the numbers of grid points in each dimension. Assuming there exists a linear TT operator $\mathbf{A}$ that describes the dynamics of the system such that

$$\mathbf{T}_k = \mathbf{A} \cdot \mathbf{T}_{k-1}, \tag{7.3.2}$$

we define the tensors $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n_1 \times \cdots \times n_d \times m}$ with

$$\mathbf{X}_{:,\ldots,:,k} = \mathbf{T}_{k-1} \quad \text{and} \quad \mathbf{Y}_{:,\ldots,:,k} = \mathbf{T}_k, \tag{7.3.3}$$

for $k = 1, \ldots, m$. The linear relationship (7.3.2) can then be expressed as

$$Y = A \cdot X, \tag{7.3.4}$$

with matricizations

$$X = \mathbf{X}\Big|^{m}_{n_1,\ldots,n_d}, \quad Y = \mathbf{Y}\Big|^{m}_{n_1,\ldots,n_d}, \quad \text{and} \quad A = \operatorname{mat}(\mathbf{A}).$$

See Section 2.4 for a description of the used notation. Let n denote the product of all modes of N, i.e. $n = n_1 \cdot \ldots \cdot n_d$. A solution of the minimization problem

$$\min_{A \in \mathbb{R}^{n \times n}} \|AX - Y\|_F$$

is given by

$$A = YX^{+}, \tag{7.3.5}$$

where $\| \, . \, \|_F$ denotes the (classical) Frobenius norm. This property of the pseudoinverse X^+ is an extension of the statement about the solution of the least-squares problem (7.1.1).

A way to analyze a given flow field is the application of DMD to the matrices X and Y. Defined as the eigenvalues and corresponding eigenvectors of the matrix A, DMD eigenvalues and modes contain dynamically relevant information about the flow field. The idea of TDMD is to extract this information directly in the TT format without reshaping the tensors $\mathbf{X}$ and $\mathbf{Y}$ into matrices. For this purpose, we use the results from the previous section, where we have shown that the pseudoinverse of X can be computed by Algorithm 6 with the tensor $\mathbf{X}$ in TT format as input.

By applying TDMD to the snapshots $\mathbf{T}_0, \mathbf{T}_1, \ldots, \mathbf{T}_m$, we consider a reduced matrix $\tilde{A}$ instead of computing A explicitly. Assume that the compact SVD of X is given by $X = U\Sigma V^T$, then it holds that $A = YV\Sigma^{-1}U^T$, see (7.1.2) and (7.1.3). We define the reduced matrix $\tilde{A}$ as

$$\tilde{A} = U^T YV\Sigma^{-1}. \tag{7.3.6}$$

Theorem 7.3.1. *The matrix A defined in* (7.3.5) *and the matrix $\tilde{A}$ defined in* (7.3.6) *share the same non-zero eigenvalues.*

Proof. If λ is an eigenvalue of A corresponding to the eigenvector v, i.e. $Av = \lambda v$, then it follows that $\tilde{A}w = \lambda w$ with $w = U^T v$. Conversely, if we have $\tilde{A}w = \lambda w$ and define $v = \frac{1}{\lambda} YV\Sigma^{-1}w$, then it holds that $Av = \lambda v$. □

There are different algorithms to compute the DMD modes of the matrix A. Here, we consider the standard DMD as well as the exact DMD algorithm. A detailed description of both DMD versions and the relation between the modes can be found in [58]. Both methods require the computation of the eigenvalues and eigenvectors of the reduced matrix $\tilde{A}$. Given an eigenvector w with $\tilde{A}w = \lambda w$, the mode corresponding to the eigenvalue λ for the standard DMD algorithm is defined as

$$\varphi = Uw. \tag{7.3.7}$$

For the exact DMD algorithm, the mode corresponding to λ is given by

$$\varphi = \frac{1}{\lambda} YV\Sigma^{-1}w. \tag{7.3.8}$$

As we already mentioned in the previous section, we do not compute the pseudoinverse X^+ explicitly. The matrix X can be expressed – after applying the steps 1 to 5 of Algorithm 6 – as $X = U\Sigma V^T$, with

$$U = \underbrace{\left(\sum_{k_0=1}^{r_0} \cdots \sum_{k_{d-1}=1}^{r_{d-1}} \mathbf{X}^{(1)}_{k_0,:,k_1} \otimes \ldots \otimes \mathbf{X}^{(d)}_{k_{d-1},:,:} \right)}_{=\mathbf{U}} \Big|^{r_d}_{n_1,\ldots,n_d}, \quad V = \underbrace{\mathbf{X}^{(d+1)}}_{=\mathbf{V}} \Big|^{r_d}_{m}, \tag{7.3.9}$$

and Σ being the diagonal matrix resulting from executing step 2 of Algorithm 6. Thus, the pseudoinverse of X is given by $X^+ = V\Sigma^{-1}U^T$. Using similar matricizations, but not requiring any special properties as left- or right-orthonormality of the TT cores, we can also represent the tensor unfolding Y as a matrix product, i.e. $Y = P \cdot Q$ with

$$P = \underbrace{\left(\sum_{l_0=1}^{s_0} \cdots \sum_{l_{d-1}=1}^{s_{d-1}} \mathbf{Y}^{(1)}_{l_0,:,l_1} \otimes \ldots \otimes \mathbf{Y}^{(d)}_{l_{d-1},:,:} \right)}_{=\mathbf{P}} \Big|^{s_d}_{n_1,\ldots,n_d}, \quad Q = \underbrace{\mathbf{Y}^{(d+1)}}_{=\mathbf{Q}} \Big|^{m}_{s_d}. \tag{7.3.10}$$

Combining the matrix representations of X^+ and Y, we can express the matrices A and $\tilde{A}$ as

$$A = Y \cdot X^+ = P\,Q \cdot V\,\Sigma^{-1}\,U^T,$$

and

$$\tilde{A} = U^T\,P\,Q\,V\,\Sigma^{-1}, \tag{7.3.11}$$

respectively, cf. (7.3.5) and (7.3.6). In order to compute $\tilde{A}$, we do not have to construct the matrices U and P explicitly. We can reduce the computational cost by splitting (7.3.11) into different parts. The product $U^TP \in \mathbb{R}^{r_d \times s_d}$ can be calculated by

$$U^TP = \langle \mathbf{U}, \mathbf{P} \rangle_{n_1,\ldots,n_d} \tag{7.3.12}$$

with the tensors $\mathbf{U}$ and $\mathbf{P}$ given in (7.3.9) and (7.3.10), respectively. Similarly, for the product $QV \in \mathbb{R}^{s_d \times r_d}$ it holds that

$$QV = \langle \mathbf{Q}, \mathbf{V} \rangle_{m}. \tag{7.3.13}$$

Using (7.3.12) and (7.3.13), we can compute both parts without leaving the TT format, i.e. we do not need to convert any tensor decomposition into full format. Given the decompositions of $\mathbf{U}$ and $\mathbf{P}$ defined in (7.3.9) and (7.3.10), respectively, the contraction (7.3.12) resembles a multiplication of two tensor trains, which can be implemented efficiently using Algorithm 4 from [42]. Afterwards, we multiply the three low-dimensional matrices $(U^T P)$, (QV), and Σ^{-1}. The latter is just a diagonal matrix containing the reciprocals of the singular values in Σ.

We can express the DMD modes using the tensor trains $\mathbf{X}$ and $\mathbf{Y}$, respectively, modifying only the last core. Assume that the eigenvalues and corresponding eigenvectors of the reduced matrix $\tilde{A}$ are given by $\lambda_1, \ldots, \lambda_c$ and $w_1, \ldots, w_c$, respectively. In order to calculate the DMD modes of A according to the standard DMD algorithm, we only replace the last TT core of $\mathbf{X}$. Considering (7.3.7), we define the matrix $W \in \mathbb{R}^{r_d \times c}$ as

$$W_{:,j} = w_j,$$

for $j = 1, \ldots, c$. The DMD modes of A can then be expressed in a TT representation $\Phi \in \mathbb{R}^{n_1 \times \cdots \times n_d \times c}$, i.e.

$$\Phi = \left[\mathbf{X}^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{X}^{(d)}\right] \otimes \left[W\right]. \tag{7.3.14}$$

It holds that

$$\varphi_j = \left(\Phi \Big|_{n_1,\ldots,n_d}^{c}\right)_{:,j}, \tag{7.3.15}$$

where φ_j is the DMD mode (in vector form) corresponding to the eigenvector w_j, see (7.3.7). Considering the exact DMD algorithm, the tensor train Φ representing all DMD modes (7.3.8) is given by

$$\Phi = \left[\mathbf{Y}^{(1)}\right] \otimes \cdots \otimes \left[\mathbf{Y}^{(d)}\right] \otimes \left[QV\Sigma^{-1}W\Lambda^{-1}\right], \tag{7.3.16}$$

with the diagonal matrix Λ containing the eigenvalues $\lambda_1, \ldots, \lambda_c$. Here, the relation (7.3.15) applies with φ_j being of the form (7.3.8).

8

Tensor-Train Approximation of the Perron–Frobenius Operator

In this chapter, we will show the first attempts to compute eigenvalues and eigentensors of finite-dimensional approximations of the so-called Perron–Frobenius operator using the (M)ALS algorithms. Other tensor-based methods for the numerical approximation of the Perron–Frobenius operator were already considered by Klus et al. in [13]. After we give the definition of the Perron–Frobenius operator, we will introduce Ulam's method, which is a frequently used method for the discretization of the Perron–Frobenius operator.

8.1. Perron–Frobenius Operator

Consider a (nonlinear) dynamical system with evolution rule $F : \mathcal{S} \to \mathcal{S}$, $\mathcal{S} \subseteq \mathbb{R}^d$. The function F describes the time-dependent evolution of a state, i.e. F gives the future state of the system (after a short time interval) following from the current state. The *Perron–Frobenius operator* $\mathcal{P}$ with $\mathcal{P} : L^2(\mathcal{S}) \to L^2(\mathcal{S})$ is defined by

$$\int_{\mathcal{S}} g(s) \cdot \mathcal{P}f(s) d\mu(s) = \int_{\mathcal{S}} (g \circ F)(s) \cdot f(s) d\mu(s), \tag{8.1.1}$$

for all $f, g \in L^2(\mathcal{S})$, see [61], where μ is a given probability measure. An eigenfunction f with eigenvalue λ of the Perron–Frobenius operator is given by

$$\mathcal{P}f = \lambda f.$$

The long-term behavior of a dynamical system can then be understood by analyzing the spectrum of $\mathcal{P}$ since the eigenfunctions can be used to decompose the dynamical system into fast and slow processes. For $\lambda = 1$, the eigenfunction f represents the invariant measure of the system. The magnitude of the second largest eigenvalue can be interpreted as the rate at which initial densities converge to the invariant measure, called the *rate of mixing* in [61]. Moreover, the leading eigenvalues with magnitude close to one correspond to decay rates associated to the slow dynamics and the eigenfunctions corresponding to these eigenvalues can be used to find almost-invariant (or metastable) sets [90].

Another important operator, which enables the analysis of the global behavior

of dynamical systems, is the *Koopman operator* [62, 91]. The Koopman operator is the adjoint of the Perron–Frobenius operator and was already considered in a tensor-based context by Klus et al. in [13]. Instead of describing the evolution of densities, the Koopman operator describes the evolution of observables, see [92]. However, we will here focus on the Perron–Frobenius operator with the intention to consider other so-called *transfer operators*, see e.g. [60], in future research. The aim is to exploit the TT format for computing finite-dimensional approximations of the eigenfunctions of the Perron–Frobenius operator and gaining insight into the system's behavior. If the eigenfunctions can be approximated accurately by low-rank TT decompositions, we may significantly reduce the required time and memory to solve the involved eigenvalue problems.

8.2. Ulam's Method

A standard method to compute finite-dimensional approximations of the Perron–Frobenius operator is *Ulam's method*, see e.g. [13, 61], which is based on the discretization of the domain $\mathcal{S}$. That is, we cover the d-dimensional domain by a finite number of disjoint boxes. If we assume that $\mathcal{S}'$ is an axis-oriented hyperrectangle in $\mathbb{R}^d$, i.e.

$$\mathcal{S}' = [a_1, b_1] \times \cdots \times [a_d, b_d] = \{(s_1, \ldots, s_d)^T \in \mathbb{R}^d : s_i \in [a_i, b_i] \text{ for } i = 1, \ldots, d\},$$

with $\mathcal{S} \subseteq \mathcal{S}'$, we partition each interval $[a_i, b_i]$ into n_i subintervals such that the boxes are given by $B(x_1, \ldots, x_d)$, $1 \leq x_i \leq n_i$. See Figure 8.1 for box discretizations of two- and three-dimensional domains.

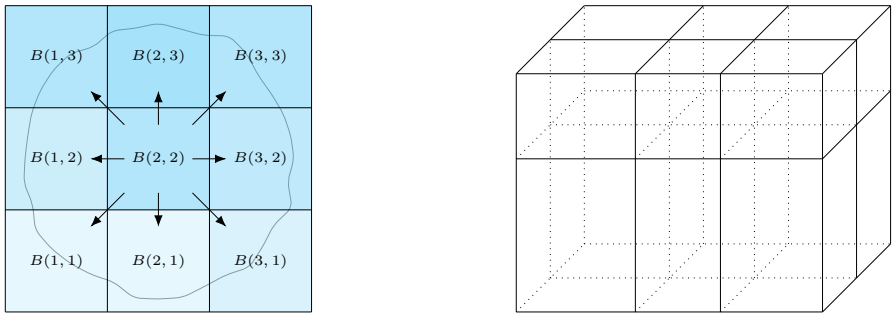


Figure 8.1: *Box discretization for Ulam's method: (a) Two-dimensional domain covered by boxes of equal size. Transition rates obtained by counting test points are visualized by the different opacities. (b) Three-dimensional domain covered by boxes of different size.*

Now, we define the indicator function for the box $B(x_1, \ldots, x_d)$ as

$$\mathbb{1}_{x_1,\ldots,x_d}(s) = \begin{cases} 1, & \text{if } s \in B(x_1, \ldots, x_d), \\ 0, & \text{otherwise.} \end{cases}$$

Equation (8.1.1) then becomes

$$\int_{\mathcal{S}} \mathbb{1}_Y(s) \cdot \mathcal{P}\mathbb{1}_X(s) d\mu(s) = \int_{\mathcal{S}} (\mathbb{1}_Y(s) \circ F)(s) \cdot \mathbb{1}_X(s) d\mu(s), \tag{8.2.1}$$

with $X = (x_1, \ldots, x_d)^T$ and $Y = (y_1, \ldots, y_d)^T$. Since the right-hand side of (8.2.1) corresponds to $\mu(F^{-1}(B(Y)) \cap B(X))$, we can express the relationship by a tensor $\mathbf{P} \in \mathbb{R}^{N \times N}$ with index set $N = (n_1, \ldots, n_d)^T \in \mathbb{N}^d$ and entries

$$\mathbf{P}_{x_1, y_1, \ldots, x_d, y_d} = \frac{\mu(F^{-1}(B(Y)) \cap B(X))}{\mu(B(X))}.$$

That is, each entry of $\mathbf{P}$ represents the probability of a point in $\mathcal{S}$ being mapped from box $B(x_1, \ldots, x_d)$ to box $B(y_1, \ldots, y_d)$ by the dynamical system.

Choosing a large ensemble of test points in $\mathcal{S}$ – per box $B(x_1, \ldots, x_d)$ we consider m randomly chosen points $s^k_{x_1, \ldots, x_d} \in \mathcal{S} \cap B(x_1, \ldots, x_d)$, $k = 1, \ldots, m$ – and applying the evolution rule F, we count how many points are mapped from $B(x_1, \ldots, x_d)$ to $B(y_1, \ldots, y_d)$. Here, we assume that all simulated particles go from one box to another and no test point is mapped outside the domain by the dynamical system.

Algorithm 7 TT approximation of the Perron–Frobenius operator (2D)

Input: List of all simulated transitions in the form of row vectors $(x_{\mu,1}, x_{\mu,2}, y_{\mu,1}, y_{\mu,2})$ for $\mu = 1, \ldots, m \cdot n_1 \cdot n_2$.

Output: Finite-dimensional TT approximation $\mathbf{P}$ of the corresponding Perron–Frobenius operator $\mathcal{P}$.

1: Find all unique combinations $(\hat{x}_1, \hat{y}_1), \ldots, (\hat{x}_r, \hat{y}_r)$ of $(x_{\mu,1}, y_{\mu,1})$, where $\mu = 1, \ldots, m \cdot n_1 \cdot n_2$ and $r \leq m \cdot n_1 \cdot n_2$.
2: Define TT core $\mathbf{P}^{(1)}$ as a tensor in $\mathbb{R}^{1 \times n_1 \times n_1 \times r}$ with all entries equal to 0.
3: **for** $\nu = 1, \ldots, r$ **do**
4: Set $\mathbf{P}^{(1)}_{1, \hat{x}_\nu, \hat{y}_\nu, \nu} = 1$.
5: **end for**
6: Define TT core $\mathbf{P}^{(2)}$ as a tensor in $\mathbb{R}^{r \times n_2 \times n_2 \times 1}$ with all entries equal to 0.
7: **for** $\mu = 1, \ldots, m \cdot n_1 \cdot n_2$ **do**
8: Set $\mathbf{P}^{(2)}_{\nu, x_{\mu,2}, y_{\mu,2}, 1}$ to $\mathbf{P}^{(2)}_{\nu, x_{\mu,2}, y_{\mu,2}, 1} + 1$, where ν is the index such that $\mathbf{P}^{(1)}_{1, x_{\mu,1}, y_{\mu,1}, \nu} = 1$.
9: **end for**
10: Set $\mathbf{P}$ to $(1/m) \cdot \mathbf{P}$.

The entries of $\mathbf{P}$ can then be estimated as

$$\mathbf{P}_{x_1,y_1,\ldots,x_d,y_d} = \frac{1}{m}\sum_{k=1}^{m} \mathbb{1}_{y_1,\ldots,y_d}\left(F\left(s^k_{x_1,\ldots,x_d}\right)\right).$$

Since the entries are normalized by the number of test points in $B(x_1,\ldots,x_d)$, the matricization of $\mathbf{P}$ is a row-stochastic matrix. Thus, the operator $\mathbf{P}$, which represents the finite-dimensional approximation of the Perron–Frobenius operator $\mathcal{P}$, defines a Markov chain on the state space $\{1,\ldots,n_1\}\times\cdots\times\{1,\ldots,n_d\}$.

For two- and three-dimensional state spaces $\mathcal{S}$, a direct construction of $\mathbf{P}$ in the TT format is shown in Algorithm 7 and 8, respectively. The input of both algorithms is a list of all observed transitions containing the box indices for each test point. That is, a transition from box $B(x_1,\ldots,x_d)$ to $B(y_1,\ldots,y_d)$ is represented by a row vector $(x_1,\ldots,x_d,y_1,\ldots,y_d)$. Note that the TT representations of the tensor operator $\mathbf{P}$ computed by Algorithm 7 and 8, respectively, may not have minimal rank, e.g. depending on the structure of $\mathbf{P}^{(2)}$, the operator $\mathbf{P}$ for two-dimensional systems may be compressed using Algorithm 15.

Algorithm 8 TT approximation of the Perron–Frobenius operator (3D)

Input: List of all simulated transitions in the form of row vectors $(x_{\mu,1},x_{\mu,2},x_{\mu,3},y_{\mu,1},y_{\mu,2},y_{\mu,3})$ for $\mu=1,\ldots,m\cdot n_1\cdot n_2\cdot n_3$.

Output: Finite-dimensional TT approximation $\mathbf{P}$ of the corresponding Perron–Frobenius operator $\mathcal{P}$.

1: Find all unique combinations $(\hat{x}_1,\hat{y}_1),\ldots,(\hat{x}_{r_1},\hat{y}_{r_1})$ of $(x_{\mu,1},y_{\mu,1})$, where $\mu=1,\ldots,m\cdot n_1\cdot n_2\cdot n_3$ and $r_1\leq m\cdot n_1\cdot n_2\cdot n_3$.
2: Define TT core $\mathbf{P}^{(1)}$ as a tensor in $\mathbb{R}^{1\times n_1\times n_1\times r_1}$ with all entries equal to 0.
3: **for** $\nu=1,\ldots,r_1$ **do**
4: Set $\mathbf{P}^{(1)}_{1,\hat{x}_\nu,\hat{y}_\nu,\nu}=1$.
5: **end for**
6: Find all unique combinations $(\hat{x}_1,\hat{y}_1),\ldots,(\hat{x}_{r_2},\hat{y}_{r_2})$ of $(x_{\mu,3},y_{\mu,3})$, where $\mu=1,\ldots,m\cdot n_1\cdot n_2\cdot n_3$ and $r_2\leq m\cdot n_1\cdot n_2\cdot n_3$.
7: Define TT core $\mathbf{P}^{(3)}$ as a tensor in $\mathbb{R}^{r_2\times n_3\times n_3\times 1}$ with all entries equal to 0.
8: **for** $\nu=1,\ldots,r_2$ **do**
9: Set $\mathbf{P}^{(3)}_{\nu,\hat{x}_\nu,\hat{y}_\nu,1}=1$.
10: **end for**
11: Define TT core $\mathbf{P}^{(2)}$ as a tensor in $\mathbb{R}^{r_1\times n_2\times n_2\times r_2}$ with all entries equal to 0.
12: **for** $\mu=1,\ldots,m\cdot n_1\cdot n_2\cdot n_3$ **do**
13: Set $\mathbf{P}^{(2)}_{\nu,x_{\mu,2},y_{\mu,2},\tilde{\nu}}$ to $\mathbf{P}^{(2)}_{\nu,x_{\mu,2},y_{\mu,2},\tilde{\nu}}+1$, where ν and $\tilde{\nu}$ are the indices such that $\mathbf{P}^{(1)}_{1,x_{\mu,1},y_{\mu,1},\nu}=1$ and $\mathbf{P}^{(3)}_{\tilde{\nu},x_{\mu,3},y_{\mu,3},1}=1$.
14: **end for**
15: Set $\mathbf{P}$ to $(1/m)\cdot\mathbf{P}$.

Moreover, the storage consumption of $\mathbf{P}$ in (sparse) TT format is slightly larger than the storage consumption of the matricization of $\mathbf{P}$ in sparse format. This is an issue, which may be resolved in future research. After expressing the tensor operator $\mathbf{P}$ in the TT format, we will use the (M)ALS algorithms for eigenvalue problems together with the BTT format – see Algorithms 13 and 14 – in order to compute the leading eigenvalues and corresponding (right-)eigentensors of $\mathbf{P}^T$. These eigentensors then approximate the eigenfunctions of the Perron–Frobenius operator.

Part III

Applications of the Tensor-Train Format

"The purpose of computing is insight, not numbers."

Richard W. Hamming,
Numerical Methods for Scientists and Engineers

In Part III of this thesis, numerical experiments from different application areas will be presented. We will give a brief overview of each topic before considering related examples. In Chapter 9, the tensor-train approach will be applied to the chemical master equation. In Chapter 10, we will consider a more general Markovian master equation describing a heterogeneous catalytic process. Numerical examples from the fields of fluid and molecular dynamics will be given in Chapters 11 and 12, respectively.

The experiments were performed on a Linux machine with 128 GB RAM and an Intel Xeon processor with a clock speed of 3 GHz and 8 cores. The algorithms were implemented in MATLAB R2015a using a compound of cell arrays and multidimensional matrices for tensors in the TT format.

9 Chemical Reaction Networks

Chemical reaction networks [93] model basic reactions between different chemical species. The aim is to analyze the time-dependent progression in order to understand and to simulate basic processes that occur in nature such as gene expression profiles and signal transduction [94, 95]. Many of these processes are described by low numbers of interacting species and chemical reactions resulting from collisions between different molecules, see e.g. [96]. In such systems, stochastic kinetics are used to describe the system dynamics since deterministic approaches are not able to include the molecular fluctuations, which play an important role for the behavior of the system, cf. [97]. After giving a brief overview of the mathematical description of chemical reaction networks, we will illustrate the efficiency of the (Q)TT approach for solving high-dimensional chemical master equations using two examples.

9.1. Elementary Reactions

In what follows, we will consider homogeneous chemical reaction networks (CRN) of constant volume V. The systems include molecules of d chemically active species S_i, $i = 1, \ldots, d$, which can react in $\mathcal{M}$ reaction channels R_μ, $\mu = 1, \ldots, \mathcal{M}$. We suppose that each reaction R_μ is an *elementary reaction*, i.e. R_μ occurs in a single reaction step without intermediate stages. A possible state of the system is represented by a vector $X = (x_1, \ldots, x_d)^T \in \mathbb{N}_0^d$, which contains the numbers of molecules of each species. Given the system in state X at time t, the fundamental hypothesis of stochastic reaction kinetics, see [80], is that the probability of a single firing of R_μ inside V in an infinitesimal time interval $[t, t + \delta t)$ is given by

$$a_\mu(X)\delta t + o(\delta t),$$

where a_μ is a positive, real valued function, called the *reaction propensity*. Here, $o(\delta t)$ denotes the Bachmann–Landau notation, i.e. $o(\delta t)$ is a function $f(\delta t)$ with $f(\delta t)/\delta t = 0$ for $\delta t \to 0$.

Let $r_{\mu,i} \in \mathbb{N}_0$ and $p_{\mu,i} \in \mathbb{N}_0$ denote the *stoichiometric coefficients* representing the number of molecules of species S_i that react and are produced due to a single firing of R_μ, respectively. We can then generally describe a CRN as the set of elementary reactions

$$\mathrm{R}_\mu : r_{\mu,1}\mathrm{S}_1 + \ldots + r_{\mu,d}\mathrm{S}_d \to p_{\mu,1}\mathrm{S}_1 + \ldots + p_{\mu,d}\mathrm{S}_d, \qquad \mu = 1, \ldots, \mathcal{M}.$$

Furthermore, we will denote the *net changes* in the number of molecules of the species caused by a single firing of R_μ by

$$\xi_\mu = (\xi_{\mu,1}, \ldots, \xi_{\mu,d})^T \in \mathbb{Z}^d,$$

with $\xi_{\mu,i} = p_{\mu,i} - r_{\mu,i}$ for $\mu = 1, \ldots, \mathcal{M}$ and $i = 1, \ldots, d$.

Elementary reactions are classified by their *molecularity*, which represents the number of reactant molecules involved in a single firing of the reaction. We will only consider *unimolecular* and *bimolecular* reactions since *termolecular* reactions are rare due to the fact that they require the collision of three particles at the same place and time, i.e. all three reactant molecules have to collide simultaneously with each other with sufficient energy. Apparently, many termolecular reactions are the combined result of two bimolecular reactions and one unimolecular reaction, cf. [98]. Furthermore, there are no known elementary reactions involving four or more molecules, see [99, 100].

Note that the notation used here is the same as in Section 5. However, in the context of chemical systems, an event is called elementary reaction and the event propensity, see (5.2.3), is called reaction propensity.

9.2. Chemical Master Equation

The fundamental equation of stochastic reaction kinetics is the *chemical master equation* (CME), which is a first-order ordinary differential equation describing the time-evolution of a CRN [80]. Unfortunately, even though the CME has relative simple structure, analytical solutions exist only for special cases since the state space grows exponentially with the number of species and therefore direct approaches are computationally infeasible for larger CRNs. A common way to simulate the system's behavior is the *stochastic simulation algorithm* [49, 101], which is based on a large ensemble of realizations of the process associated with the CME. Hence, if some events in the system are rare, we need a large number of realizations to ensure a sufficient sampling. Using the results from Section 5, the approach we propose in this work is to solve the CME directly by using the TT format (or QTT format) in order to mitigate the curse of dimensionality. Similar approaches can be found in [4] and [5].

For a given initial state $X_0 \in \mathbb{N}_0^d$ at time $t_0 \in \mathbb{R}$, we denote the probability that the CRN is in state $X \in \mathbb{N}_0^d$ at time $t \geq t_0$ by $P(X, t)$. As it was done in Section 5, we again omit the dependence on the initial state X_0 in the notation for the sake of simplicity. The CME is given by

$$\frac{\partial}{\partial t} P(X,t) = \sum_{\mu=1}^{M} \left(a_\mu(X - \xi_\mu) P(X - \xi_\mu, t) - a_\mu(X) P(X,t) \right). \tag{9.2.1}$$

If $(X - \xi_\mu) \notin \mathbb{N}_0^d$ for a $\mu \in \{1, \ldots, M\}$, we set $a_\mu(X - \xi_\mu) = 0$ and $P(X - \xi_\mu, t) = 0$.

Equation (9.2.1) can be directly derived from the fundamental hypothesis, see [80]. We assume that the probability distribution vanishes outside a bounded domain and therefore is negligible above a certain number of molecules. Therefore, we truncate the considered state space to a finite domain, cf. [102]. That is, we assume $0 \leq x_i \leq n_i \in \mathbb{N}$ for $i = 1, \ldots, d$. The CME (9.2.1) was already given in Section 5 as a special case of MMEs. However, the state space for chemical systems is of a slightly different structure than it was the case for Markov processes in general, cf. (5.2.1). Representing the number of molecules, the state space $\mathcal{S}$ is given by

$$\mathcal{S} = \{0, \ldots, n_1\} \times \{0, \ldots, n_2\} \times \cdots \times \{0, \ldots, n_d\}, \tag{9.2.2}$$

For a state vector $X = (x_1, \ldots, x_d) \in \mathcal{S}$, we again associate tensors with the functions P and a_μ, i.e. we define

$$(\mathbf{P}(t))_{x_1+1,\ldots,x_d+1} = P(X, t),$$

and

$$(\mathbf{a}_\mu)_{x_1+1,\ldots,x_d+1} = a_\mu(X).$$

Note that the indices of $\mathbf{P}$ and $\mathbf{a}_\mu$ start at 1, but each x_i can be 0 for $i = 1, \ldots, d$. As a result, it holds that $\mathbf{P}, \mathbf{a}_\mu \in \mathbb{R}^{n_1+1 \times \ldots \times n_d+1}$. Thus, truncating the state space and using the relations above, we can write down the tensor-based counterpart of (9.2.1) as

$$\frac{\partial}{\partial t}\mathbf{P}(t) = \left(\sum_{\mu=1}^{\mathcal{M}} (\mathbf{G}_\mu - \mathbf{I}) \cdot \operatorname{diag}(\mathbf{a}_\mu) \right) \cdot \mathbf{P}(t) = \mathbf{A} \cdot \mathbf{P}(t), \tag{9.2.3}$$

where $\mathbf{G}_\mu$ are the multidimensional shift operators given in Definition 5.2.1 and $\mathbf{I}$ is the identity tensor, see Example 2.1.1. The diagonalized propensities $\operatorname{diag}(\mathbf{a}_\mu)$ are given by (5.2.6) and (5.2.8).

9.3. Numerical Experiments

9.3.1. Signaling Cascade

The first example is a cascading process on a genetic network consisting of genes of species $\mathrm{S}_1, \ldots, \mathrm{S}_d$. As we described in [9], the system can be expressed as an NNIS, see Section 6.1, where the cells represent the adjacent genes and a state of a cell corresponds to the number of proteins. The structure of this system is shown in Figure 9.1. The reactions and corresponding reaction propensities for a state $X = (x_1, \ldots, x_d) \in \mathbb{N}_0^d$ are:

Creation of the first protein corresponding to species S_1:

$$\mathrm{R}_1 : \varnothing \to \mathrm{S}_1, \quad a_1(X) = 0.7.$$

Creation of a protein corresponding to species S_i, $2 \leq i \leq d$:

$$\mathrm{R}_i : \mathrm{S}_{i-1} \to \mathrm{S}_{i-1} + \mathrm{S}_i, \quad a_i(X) = \frac{x_{i-1}}{5 + x_{i-1}}.$$

Destruction of a protein corresponding to species S_i, $1 \leq i \leq d$:

$$\mathrm{R}_{i+d} : \mathrm{S}_i \to \varnothing, \quad a_{i+d}(X) = 0.07\, x_i.$$

Cascading processes defined by the creation and destruction reactions above have already been considered using different methods. In [103], the 3-dimensional case was treated using a sparse grid technique. Later, the cascade model with 20 genes was analyzed using a greedy algorithm in the canonical format, see [104]. A tensor-train approach was introduced in [5], where the chemical master equation corresponding to the 20-dimensional problem was solved numerically using a simultaneous space-time discretization approach. Here, we will simply employ the trapezoidal rule as described in Section 4.5 in order to compute the numerical solution of the corresponding MME within the time interval $[0, 300]$.

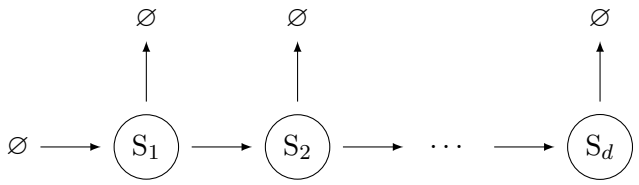


Figure 9.1: *Visualization of the signaling cascade: The genes of species* $\mathrm{S}_1, \ldots, \mathrm{S}_d$ *produce proteins affecting the expression of subsequent genes. Additionally, the produced proteins are destructed over time.*

Starting with an initial state where all numbers of proteins are zero, the probability density function for any $x_i \geq 63$ is below machine precision for all times $t \geq 0$ and $i = 1, \ldots, d$, see [5]. Therefore, we consider a finite state space

$$\mathcal{S} = \{0, \ldots, 63\} \times \cdots \times \{0, \ldots, 63\}.$$

The corresponding NNIS is non-cyclic and heterogeneous since the first creation reaction differs from the other creation reactions. In [5], one can find an exact TT decomposition of the MME operator of this system. However, the system can be represented using a SLIM decomposition, which we presented in [9]. Written as

rank-one tensors, the reaction propensities have the form

$$\mathbf{a}_1 = 0.7 \cdot \mathbb{1} \otimes \cdots \otimes \mathbb{1}, \qquad \mathbf{a}_{d+1} = \begin{pmatrix} 0.07 \cdot 0 \\ \vdots \\ 0.07 \cdot 63 \end{pmatrix} \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1},$$

$$\mathbf{a}_2 = \begin{pmatrix} \frac{0}{5+0} \\ \vdots \\ \frac{63}{5+63} \end{pmatrix} \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1}, \qquad \mathbf{a}_{d+2} = \mathbb{1} \otimes \begin{pmatrix} 0.07 \cdot 0 \\ \vdots \\ 0.07 \cdot 63 \end{pmatrix} \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1},$$

$$\vdots \qquad\qquad \vdots$$

$$\mathbf{a}_d = \mathbb{1} \otimes \cdots \otimes \mathbb{1} \otimes \begin{pmatrix} \frac{0}{5+0} \\ \vdots \\ \frac{63}{5+63} \end{pmatrix} \otimes \mathbb{1}, \qquad \mathbf{a}_{2d} = \mathbb{1} \otimes \cdots \otimes \mathbb{1} \otimes \begin{pmatrix} 0.07 \cdot 0 \\ \vdots \\ 0.07 \cdot 63 \end{pmatrix},$$

where $\mathbb{1} = (1, \ldots, 1) \in \mathbb{R}^{64}$. The vectors of net changes are all zero except for one entry, i.e.

$$\xi_1 = \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}, \qquad \xi_{d+1} = \begin{pmatrix} -1 & 0 & \cdots & 0 \end{pmatrix},$$

$$\vdots \qquad\qquad \vdots$$

$$\xi_d = \begin{pmatrix} 0 & \cdots & 0 & 1 \end{pmatrix}, \qquad \xi_{2d} = \begin{pmatrix} 0 & \cdots & 0 & -1 \end{pmatrix}.$$

Now, for the sake of simplicity, we define the shift matrices $G^{\downarrow} := G_i(-1)$ and $G^{\uparrow} := G_i(1)$ for $i = 1, \ldots, d$, i.e.

$$G^{\downarrow} = \begin{pmatrix} 0 & & & 0 \\ 1 & 0 & & \\ & \ddots & \ddots & \\ 0 & & 1 & 0 \end{pmatrix} \qquad \text{and} \qquad G^{\uparrow} = \begin{pmatrix} 0 & 1 & & 0 \\ & 0 & \ddots & \\ & & \ddots & 1 \\ 0 & & & 0 \end{pmatrix}, \tag{9.3.1}$$

cf. Section 5.2. The corresponding shift operators for the creation and destruction reactions are then given by

$$\mathbf{G}_1 = G^{\downarrow} \otimes I \otimes \cdots \otimes I, \quad \ldots, \quad \mathbf{G}_d = I \otimes \cdots \otimes I \otimes G^{\downarrow},$$

and

$$\mathbf{G}_{d+1} = G^{\uparrow} \otimes I \otimes \cdots \otimes I, \quad \ldots, \quad \mathbf{G}_{2d} = I \otimes \cdots \otimes I \otimes G^{\uparrow}.$$

In the canonical format, we can now express the MME operator as

$$\begin{aligned}\mathbf{A} = 0.7 \cdot G^{\downarrow} \otimes I \otimes \cdots \otimes I \;\; &- \;\; 0.7 \cdot I \otimes I \otimes \cdots \otimes I \\ + \;\; H_1 \otimes G^{\downarrow} \otimes I \otimes \cdots \otimes I \;\; &- \;\; H_1 \otimes I \otimes \cdots \otimes I \\ + \;\; &\ldots \\ + \;\; I \otimes \cdots \otimes I \otimes H_1 \otimes G^{\downarrow} \;\; &- \;\; I \otimes \cdots \otimes I \otimes H_1 \otimes I \\ + \;\; (G^{\uparrow} \cdot H_2) \otimes I \otimes \cdots \otimes I \;\; &- H_2 \otimes I \otimes \cdots \otimes I \\ + \;\; &\ldots \\ + \;\; I \otimes \cdots \otimes I \otimes (G^{\uparrow} \cdot H_2) \;\; &- \;\; I \otimes \cdots \otimes I \otimes H_2,\end{aligned}$$

with identity matrix $I \in \mathbb{R}^{64\times 64}$ and

$$H_1 = \operatorname{diag}\left(\frac{0}{5}, \frac{1}{6}, \ldots, \frac{63}{68}\right), \qquad H_2 = 0.07 \cdot \operatorname{diag}(0, 1, \ldots, 63),$$

where $\operatorname{diag}(v)$ denotes the square diagonal matrix with the elements of the vector $v \in \mathbb{R}^{64}$ on the main diagonal. By defining

$$\begin{gathered}\mathbf{S}^* = 0.7 \cdot \left(G^{\downarrow} - I\right), \quad \mathbf{S} = \left(G^{\uparrow} - I\right) \cdot H_2, \\ \mathbf{L} = H_1, \quad \mathbf{I} = I, \quad \mathbf{M} = G^{\downarrow} - I,\end{gathered}$$

we obtain the SLIM decomposition

$$\mathbf{A} = \begin{bmatrix}\mathbf{S}^* & \mathbf{L} & \mathbf{I}\end{bmatrix} \otimes \begin{bmatrix}\mathbf{I} & 0 & 0 \\ \mathbf{M} & 0 & 0 \\ \mathbf{S} & \mathbf{L} & \mathbf{I}\end{bmatrix} \otimes \cdots \otimes \begin{bmatrix}\mathbf{I} & 0 & 0 \\ \mathbf{M} & 0 & 0 \\ \mathbf{S} & \mathbf{L} & \mathbf{I}\end{bmatrix} \otimes \begin{bmatrix}\mathbf{I} \\ \mathbf{M} \\ \mathbf{S}\end{bmatrix}, \tag{9.3.2}$$

which has TT ranks equal to 3 for any number of species.

Considering a network of $d = 20$ genes, we use this example to demonstrate the advantage of the QTT approach over a direct formulation of the system in TT format. We quantize the state space, i.e. the operator $\mathbf{A} \in \mathbb{R}^{M\times N}$ with index sets $M = N = (64, \ldots, 64)^T \in \mathbb{N}^{20}$ is further decomposed into a QTT operator $\tilde{\mathbf{A}} \in \mathbb{R}^{\tilde{M}\times\tilde{N}}$ with the index sets $\tilde{M} = \tilde{N} = (2, \ldots, 2)^T \in \mathbb{N}^{120}$ by using Algorithm 3. That is, each TT core of $\mathbf{A}$ is split into six QTT cores. The result is then an operator with QTT ranks bounded by 12. Even though some of the QTT ranks of $\tilde{\mathbf{A}}$ are higher than the TT ranks of $\mathbf{A}$, we can reduce the computational effort for solving the MME significantly by quantizing the state space.

Table 9.1.: *Solving the cascade problem in the TT format: Maximum relative errors and CPU times depending on the bound of the TT ranks of the initial guess. The bound is denoted as TT rank.*

TT rank	$\max_k\{e_{\mathrm{TR},k}\}$	CPU time
1	1.73e−01	9.86 s
2	1.38e−01	24.62 s
3	8.77e−02	74.95 s
4	5.90e−02	207.26 s
5	4.71e−02	497.60 s
6	2.43e−02	1422.10 s

For both the TT and QTT approach, we fix the step size to $\tau = 1$ for all 300 steps and start with an initial distribution $\mathbf{P}_0$ with

$$(\mathbf{P}_0)_{x_1,x_2,\ldots,x_{20}} = \begin{cases} 1, & \text{if } x_1 = x_2 = \cdots = x_{20} = 1, \\ 0, & \text{otherwise,} \end{cases}$$

such that the concentration of each species is zero at the beginning. The ALS algorithm, see Section 4.2, is used for solving the systems of linear equations of the form (4.5.4) at each iteration step of the implicit Euler method. As an initial guess for the first system of linear equations, we define a uniformly distributed tensor, i.e. all entries of the initial guess are equal and sum up to 1. Subsequently, the computed distribution after one time step is then used as initial guess for the next system of linear equations. The resulting tensors $\mathbf{P}_k$, $k = 1, \ldots, 300$, representing the probability distributions over all states at times $t_k = k \cdot \tau$ are then normalized such that $\|\mathbf{P}_k\|_1 = 1$, assuming that the absolute values of potentially negative entries are small enough to be omitted, see Section 3.4.4. Note that the MME operator (9.3.2) is non-symmetric and therefore the systems of linear equations of the form (4.5.4) are non-symmetric. Since the operators of the corresponding normal equations, cf. Section 4.4, are extremely ill-conditioned, we apply ALS to the non-symmetric systems directly, cf. [5]. After computing the numerical solution, we calculate the relative errors $e_{\mathrm{TR},k}$ of the systems of linear equations at each iteration step, see (4.5.5), in order to estimate the accuracy. Tables 9.1 and 9.2 show the maximum relative errors and the CPU times depending on the ranks for the TT and the QTT approach, respectively. The results show that we obtain a better accuracy while reducing the required CPU time, even with higher ranks. For instance, in order to get an error bound of approximately 2.5%, we only need around 105 s to compute the numerical solution of the MME in QTT format with rank bound 11. In contrast to that, the CPU time needed for a comparable accuracy using the TT approach is nearly 14 times larger. Additionally, the computation times increase at a much lower rate for the QTT approach than for the direct TT approach.

Table 9.2.: *Solving the cascade problem in the QTT format: Maximum relative errors and CPU times depending on the bound of the QTT ranks of the initial guess. The bound is denoted as QTT rank.*

QTT rank	$\max_k\{e_{\mathrm{TR},k}\}$	CPU time
2	3.73e−01	25.68 s
4	1.50e−01	28.10 s
6	6.99e−02	52.09 s
8	3.81e−02	77.87 s
10	2.20e−02	103.79 s
12	1.64e−02	151.23 s

After computing the numerical solution of the MME, we are interested in the mean concentrations of all species over time. The average number of molecules of species S_i at time t_k is given by

$$\overline{x_i}(t_k) = \sum_{X \in \mathcal{S}} x_i \cdot (\mathbf{P}_k)_{x_1+1,\ldots,x_d+1} \,. \tag{9.3.3}$$

Figure 9.2 (a) shows the mean concentrations over the time interval $[0, 300]$. Note the time delay between equal concentrations of the different species, which is typical for cascading processes. We estimate the closeness to the stationary distribution, see Figure 9.2 (b), in terms of the right-hand side of (9.2.3) since $\|\mathbf{A} \cdot \mathbf{P}_k\|_2$ should be close to 0 for accurate approximations of the stationary distribution.

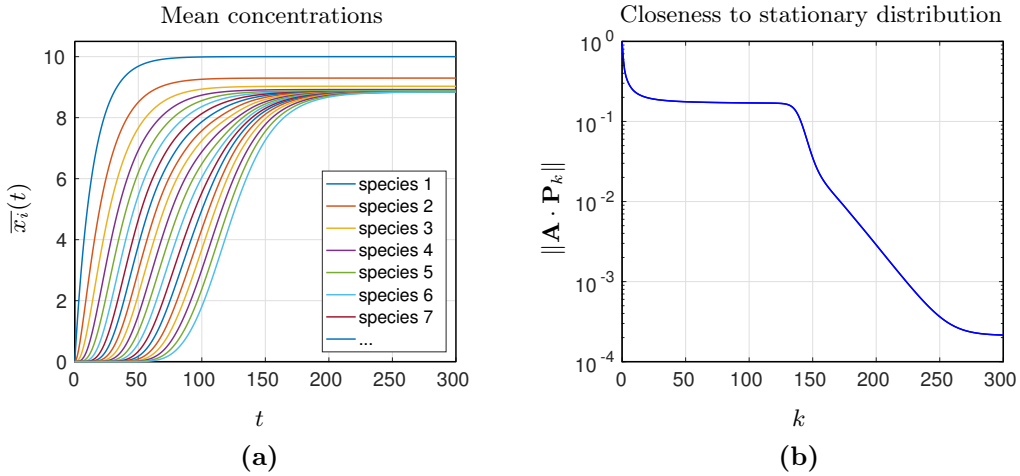


Figure 9.2: *Results for the 20-dimensional signaling cascade: (a) Mean concentrations over time. (b) Closeness to the stationary distribution. The values are calculated from the numerical solution obtained by the QTT approach with rank bound 12. The same qualitative behavior can be observed when using a direct TT formulation of the problem.*

As one can see in Figure 9.2 (b), the norms of $\mathbf{A} \cdot \mathbf{P}_k$ stagnate at a level of around 0.17 in the first part of the time interval and decrease rapidly within the second part. The location of the kink approximately corresponds to the changing behavior of the mean concentrations plotted in Figure 9.2 (a). When the average numbers of molecules are close to the steady state for each species, the tensor approximations of the probability distributions converge faster to the stationary distribution.

9.3.2. Two-Step Destruction

As a second example, we consider a two-step mechanism where molecules of a certain species S_1 react with molecules of another species S_2 and form an intermediate species. After a second reaction with S_2, the resulting product then vanishes in time. Realistic examples of chemical reactions based on two-step mechanisms are the reaction between iodine monochloride and hydrogen [105, 106] as well as the reaction between carbon monoxide and nitrogen dioxide [107, 108]. In order to model the destruction process, we consider a CRN including four species $\mathrm{S}_1, \ldots, \mathrm{S}_4$ and three reactions defined as

$$\begin{array}{lrcl} \mathrm{R}_1: & \mathrm{S}_1 + \mathrm{S}_2 & \to & \mathrm{S}_3 \\ \mathrm{R}_2: & \mathrm{S}_3 + \mathrm{S}_2 & \to & \mathrm{S}_4 \\ \mathrm{R}_3: & \mathrm{S}_4 & \to & \varnothing \end{array}$$

A visualization of the system is shown in Figure 9.3. Following the *rate law* for chemical reactions, see e.g. [106], we assume that the corresponding reaction propensities are given by $a_1(X) = k_1 x_1 x_2$, $a_2(X) = k_2 x_2 x_3$ and $a_3(X) = k_3 x_4$ with *rate constants* $(k_1, k_2, k_3) = (1, 2, 1)$ and $X = (x_1, x_2, x_3, x_4)^T$ being the state of the system. We consider a finite state space $\mathcal{S}$ of the form (9.2.2), where $n_1 = n_3 = n_4 = 2^m - 1$ and $n_2 = 2^{m+1} - 1$.

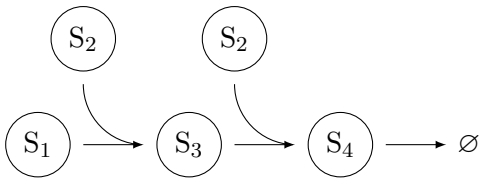


Figure 9.3: *Visualization of the two-step destruction process: After two consecutive bimolecular reactions with molecules of species* S_2*, the final products degrade over time. Note that no molecules of species* S_1 *and* S_2*, respectively, are created.*

The CME operator, see (9.2.3), of this system is then given by

$$\mathbf{A} = \sum_{\mu=1}^{3} (\mathbf{G}_\mu - \mathbf{I}) \cdot \mathrm{diag}(\mathbf{a}_\mu),$$

with

$$\mathbf{G}_1 = \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & & & \\ 1 & \ddots & & \\ & \ddots & 0 & \\ & & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix},$$

$$\mathbf{G}_2 = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & & & \\ 1 & \ddots & & \\ & \ddots & 0 & \\ & & 1 & 0 \end{pmatrix},$$

$$\mathbf{G}_3 = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix},$$

and

$$\mathbf{a}_1 = k_1 \cdot \begin{pmatrix} 0 \\ 1 \\ \vdots \\ n_1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ \vdots \\ n_2 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix},$$

$$\mathbf{a}_2 = k_2 \cdot \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ \vdots \\ n_2 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ \vdots \\ n_3 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix},$$

$$\mathbf{a}_3 = k_3 \cdot \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ \vdots \\ n_4 \end{pmatrix},$$

see Chapter 5 for details regarding the construction.

In the TT format, the tensor operator $\mathbf{A}$ can be expressed as

$$\mathbf{A} = \begin{bmatrix} I_1 & -D_1 & G_1^{\uparrow} \cdot D_1 \end{bmatrix} \otimes \begin{bmatrix} G_2^{\uparrow} \cdot D_2 & I_2 & -D_2 & 0 & 0 \\ 0 & 0 & 0 & D_2 & 0 \\ 0 & 0 & 0 & 0 & G_2^{\uparrow} \cdot D_2 \end{bmatrix} \otimes \begin{bmatrix} G_3^{\uparrow} \cdot D_3 & 0 & 0 \\ 0 & I_3 & 0 \\ 0 & 0 & D_3 \\ 0 & 0 & I_3 \\ 0 & 0 & G_3^{\downarrow} \end{bmatrix} \otimes \begin{bmatrix} G_4^{\downarrow} \\ G_4^{\uparrow} \cdot D_4 - D_4 \\ I_4 \end{bmatrix}, \tag{9.3.4}$$

where $I_i, D_i, G_i^{\uparrow}, G_i^{\downarrow} \in \mathbb{R}^{(n_i+1)\times(n_i+1)}$ with $I_i = I$, $D_i = \operatorname{diag}(0, \ldots, n_i)$, and the shift matrices $G_i^{\uparrow}$ and $G_i^{\downarrow}$ of the form (9.3.1).

In order to illustrate the efficiency of the tensor approach, we set the exponent m to natural numbers between 2 and 5. That is, the number of states we consider in this experiment reaches from 2^9 to 2^{21}, which is rather small compared to the state space from the previous experiment. However, the following example shows that the efficiency of the tensor approach strongly depends on the structure of the given system, i.e. the amount of states with non-vanishing probabilities is more crucial than the total number of states.

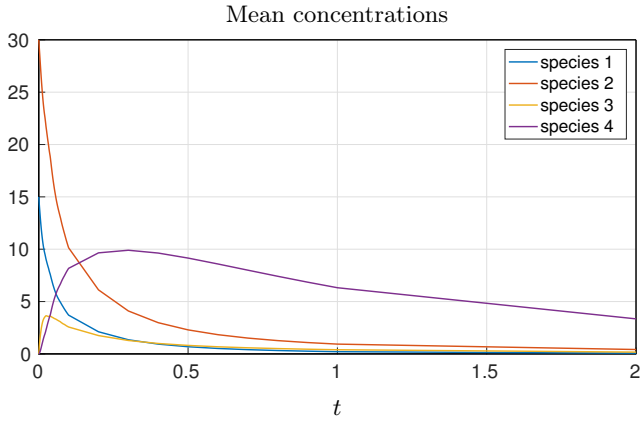


Figure 9.4: *Mean concentrations for the two-step destruction: The average numbers of molecules of the different species change rapidly at the beginning and then slowly converge to the steady state of all molecules being vanished.*

We apply Algorithm 3 again and decompose the TT cores of (9.3.4) into m and $m+1$ QTT cores, respectively. All mode sizes of the quantized operator are then equal to 2. Starting with an initial state $X_0 = (2^m - 1, 2^{m+1} - 2, 0, 0)$, we can ensure that the probabilities for states outside the state space $\mathcal{S}$ are 0 since no molecules of S_1 are produced and the maximum number of molecules of S_3 and S_4, respectively, is bounded by the maximum number of molecules of S_1. In order to simulate the conversion of all molecules, we set the initial number of molecules of S_2 to $2n_1 = 2^{m+1} - 2$.

Table 9.3.: *Solving the destruction problem in the QTT format: Maximum relative errors and CPU times depending on the exponent m determining the size of the state space $\mathcal{S}$.*

m	$\max_k\{e_{\mathrm{IE},k}\}$	CPU time
2	2.15e−02	1.47 s
3	8.57e−02	64.12 s
4	1.32e−01	431.83 s
5	1.31e+00	834.67 s

We employ the implicit Euler method in combination with MALS, see Chapter 4, to compute the numerical solution of the MME up to $t = 10$. Here, we let MALS adapt the TT ranks of the probability tensors during the computation, but only allow TT ranks not greater than 30. Starting with a step size of 0.001, we compute the numerical solution on the time interval $[0, 0.1]$. As we observed in our experiments, the probability distribution over the state space changes rather slowly after this time interval. Thus, we then compute 9 steps each with step sizes 0.1 and 1, respectively. Table 9.3 shows the CPU times and maximum approximation errors of the form (4.5.3) depending on the exponent m. The error increases with growing m since the rank bound for MALS causes a loss in accuracy. In particular, higher QTT ranks would be needed for approximating the transient process corresponding to values of m larger than 4.

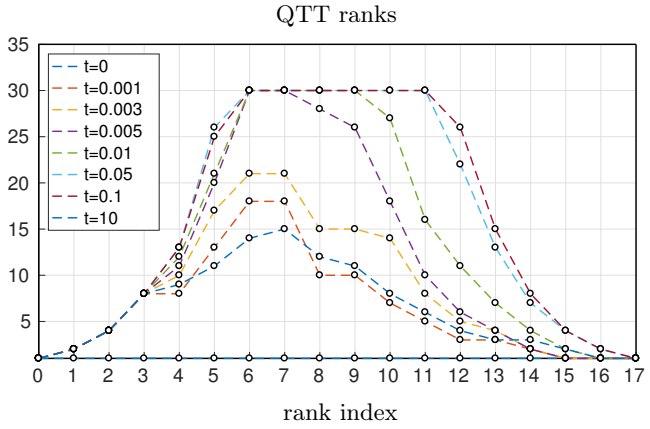


Figure 9.5: *QTT ranks for the two-step destruction: The adaption of the QTT ranks by MALS corresponds to the density of the probability distributions. When the number of states with non-vanishing probability increases, higher QTT ranks are needed for the approximation.*

For $m = 4$, the mean concentrations of the species over time as given in (9.3.3) are shown in Figure 9.4. We only show the varying concentrations over the time interval $[0, 2]$. After $t = 2$, the first three species are almost fully converted and the number of molecules of species S_4 is monotonically decreasing. The same qualitative behavior can be observed for other values of m, even roughly for $m = 5$. After the probability distribution changes rapidly within the time interval $[0, 1]$, the QTT approximations slowly converge to the stationary distribution, which is also reflected in the QTT

ranks of the approximations. The QTT ranks naturally decrease when the numerical solution gets closer to the steady state, which can be represented as a rank-one tensor where the probability is concentrated in the state $X = (0, 0, 0, 0)^T \in \mathcal{S}$. Figure 9.5 shows the QTT ranks of the probability distributions at certain time steps. In accordance with the convergence behavior observed in Figure 9.4, the QTT ranks for $m = 4$ increase rapidly at the beginning and then start to decrease after $t \approx 2$. At $t = 10$, almost all molecules (and therefore the probabilities for most states) are vanished such that the distribution at that point can be represented by a tensor with QTT ranks bounded by 15.

10
Heterogeneous Catalysis

This chapter is based on our work in [6] and [9]. We will show how to exploit the TT format in order to mitigate the curse of dimensionality for a reduced model of the oxidation of carbon monoxide on a catalytic surface, see [109, 110, 111]. A central objective of heterogeneous catalytic processes is to investigate the stationary behavior of a catalyst. Therefore, our experiments will focus on computing stationary probability distributions. After giving a brief introduction of heterogeneous catalytic processes and describing the considered model, we will examine the computational complexity for increasing system size and for various reaction conditions in Section 10.3.1. Furthermore, we will compare the results obtained by the TT approach with kinetic Monte Carlo simulations in Section 10.3.2 and illustrate the efficiency of the TT approach in Section 10.3.3 by considering a numerical example with increasing stiffness.

10.1. Heterogeneous Catalytic Processes

Catalysts are substances that facilitate chemical reactions without being consumed in the process. Heterogeneous catalysts are usually solid materials while the reactants are gases or liquids. These reactants are adsorbed onto the surface of the catalyst at so-called *active sites*. Apart from the actual chemical reactions on the catalytic surface, further possible events are the diffusion of the reactant molecules as well as the desorption of the products. Processes where heterogeneous catalysis plays an important role range from artificial photo-synthesis [54] to automotive exhaust gas cleaning [55]. More examples for the economic importance of heterogeneous catalysts can be found in [112, 113].

When modeling heterogeneous catalytic processes, direct molecular dynamics simulations of rare-event systems are limited in the sampling they can achieve since chemical reactions are – on a time scale of molecular motion – rare transitions from one metastable basin to another [114]. However, due to rapid motion within each basin, we can assume that the next transition only depends on the current metastable state. Thus, if we only consider the sequence of these metastable states, the coarse-grained dynamics can be modeled as a Markov jump process, see Section 5.1, where each jump corresponds to the execution of a particular reaction event. The state space of these processes is in general extremely high-dimensional such that standard numerical techniques cannot be applied. Assuming negligible correlations between species at different active sites, a simple approach to evaluate the surface kinetics is the *mean-field approximation*. However, when applied

to general many-body systems, mean-field approximations might lead to inaccurate results [110, 115, 116]. A rather modern approach to analyze the chemical kinetics is the application of the *kinetic Monte Carlo* (kMC) method [50, 51, 117], also known as *stochastic simulation algorithm*, cf. Section 9.2. The drawback of the kMC method is the large number of simulations needed to capture the relevant dynamics. Thus, there is a demand for methods which can overcome the disadvantages of those methods while maintaining the scaling behavior of the computational complexity with growing number of dimensions.

10.2. Reduced Model for the CO Oxidation at RuO_2

Ruthenium dioxide ($RuO_2(110)$, where (110) is the *index*, see [118]) is a promising catalyst for different oxidation reactions [119]. Here, we consider the oxidation of carbon monoxide (CO) at the surface. Given gaseous oxygen (O_2) and CO as reactants binding to the $RuO_2(110)$ surface, carbon dioxide (CO_2) is produced and desorbed from the surface.

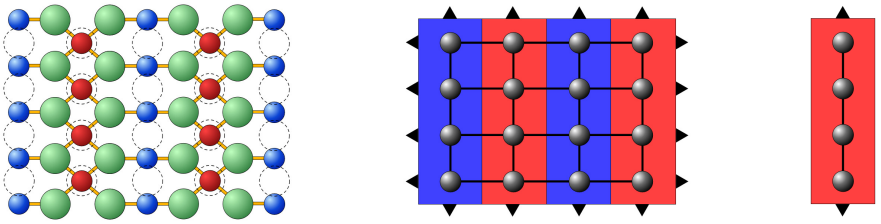


Figure 10.1: *Reduced model for the CO oxidation at $RuO_2(110)$: (a) Top view of the $RuO_2(110)$ surface showing the two prominent adsorption sites, bridge sites between the ruthenium atoms in blue and cus sites on the ruthenium atoms in red. (b) Two-dimensional lattice model of the coarse-grained surface composed of alternating rows of bridge and cus sites. (c) One-dimensional lattice model completely composed of cus sites.*

The considered microkinetic model for the CO oxidation at $RuO_2(110)$ was presented by Reuter et al. in 2004 [109, 120]. It is derived by using *density-functional theory* [121] together with *transition-state theory* [122] and reproduces experimental findings reasonably well. A top view of the surface model is shown in Figure 10.1 (a). Considering a rectangular lattice with alternating columns of so-called *bridge sites* and *coordinatively unsaturated sites* (cus), see Figure 10.1 (b), we are interested in the limit of very large lattices since the typical lattice spacing is very small (a few Ångström) compared to the size of the catalyst. Thus, periodic boundaries will be employed to mimic an infinite system. Each adsorption site has three different states:

$$1 \mathrel{\widehat{=}} \text{empty}, \quad 2 \mathrel{\widehat{=}} \text{O-covered}, \quad 3 \mathrel{\widehat{=}} \text{CO-covered}.$$

The possible reaction events are:

- unimolecular adsorption of CO on bridge and cus sites, respectively,
- unimolecular desorption of CO on bridge and cus sites, respectively,
- dissociative adsorption of O_2 on two neighboring sites of any kind,
- associative desorption of O_2 from two neighboring sites of any kind,
- diffusion of CO and O, respectively, to a neighboring site of any kind,
- associative desorption of CO_2 from neighboring sites of any kind.

We assume the same environmental conditions as in [110]. In detail, these are a fixed O_2 pressure of $p_{O_2} = 1$ atm, a fixed temperature $T = 600$ K, and a varying CO pressure $p_{CO} \in [10^{-4}, 10^{2}]$ atm. This set of gas-phase conditions is representative for so-called *in-situ experiments*, i.e. the conditions are close to the operation conditions for a catalyst in a realistic scenario. As the rate of CO adsorption depends linearly on the CO pressure, the interval for p_{CO} corresponds to $k_{CO}^{Ad} \in [10^4, 10^{10}]\,\mathrm{s}^{-1}$.

In [110, 123], it has been found that the chemical kinetics predominantly take place only on the cus sites. Thus, we omit all reactions involving bridge sites and restrict to a reduced model of non-communicating columns of cus sites. In this way, the problem becomes one-dimensional in the form of a ring consisting of d cus sites, see Figure 10.1 (c). Table 10.1 summarizes the elementary reactions and corresponding rate constants.

Table 10.1.: *Elementary reaction steps and corresponding rate constants: The reactions are defined on two neighboring cus sites Θ_i and Θ_j, except for adsorption and desorption of CO, which are defined only on one cus site Θ_i.*

Adsorption								
$R_{O_2}^{Ad}$	:	$\varnothing_i + \varnothing_j$	$\rightarrow$	$O_i + O_j$	,	$k_{O_2}^{Ad}$	=	$9.7 \cdot 10^7 s^{-1}$
R_{CO}^{Ad}	:	$\varnothing_i$	$\rightarrow$	CO_i	,	k_{CO}^{Ad}	=	$10^4 - 10^{10} s^{-1}$
Desorption								
$R_{O_2}^{De}$	:	$O_i + O_j$	$\rightarrow$	$\varnothing_i + \varnothing_j$	,	$k_{O_2}^{De}$	=	$2.8 \cdot 10^1 s^{-1}$
R_{CO}^{De}	:	CO_i	$\rightarrow$	$\varnothing_i$	,	k_{CO}^{De}	=	$9.2 \cdot 10^6 s^{-1}$
$R_{CO_2}^{De}$	:	$CO_i + O_j$	$\rightarrow$	$\varnothing_i + \varnothing_j$	,	$k_{CO_2}^{De}$	=	$1.7 \cdot 10^5 s^{-1}$
Diffusion								
R_{O}^{Diff}	:	$O_i + \varnothing_j$	$\rightarrow$	$\varnothing_i + O_j$	,	k_{O}^{Diff}	=	$0.5 s^{-1}$
R_{CO}^{Diff}	:	$CO_i + \varnothing_j$	$\rightarrow$	$\varnothing_i + CO_j$	,	k_{CO}^{Diff}	=	$6.6 \cdot 10^{-2} s^{-1}$

The first time, we considered the CO oxidation at a $RuO_2(110)$ surface in the context of TT decompositions, see [6], we constructed the corresponding master equation operator (5.2.9) by hand. However, as we later showed in [9], the system is a cyclic, homogeneous NNIS, see Section 6.1, and therefore can be expressed as a SLIM decomposition given in (6.2.4). The cells $\Theta_1, \ldots, \Theta_d$ then represent the adsorption sites on the surface. In order to construct the operator corresponding to the MME, we use Algorithm 16 with inputs

$$
\begin{aligned}
\mathrm{a}_{i,1} &= \begin{pmatrix} k_{\mathrm{CO}}^{\mathrm{Ad}} & 0 & 0 \end{pmatrix}, & p_{i,1} &= +2,\\
\mathrm{a}_{i,2} &= \begin{pmatrix} 0 & 0 & k_{\mathrm{CO}}^{\mathrm{De}} \end{pmatrix}, & p_{i,2} &= -2,\\
\mathrm{a}_{i,i+1,1} &= \begin{pmatrix} k_{\mathrm{O}_2}^{\mathrm{Ad}} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, & [p_{i,i+1,1},\, q_{i,i+1,1}] &= [+1,\, +1],\\
\mathrm{a}_{i,i+1,2} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & k_{\mathrm{O}_2}^{\mathrm{De}} & 0 \\ 0 & 0 & 0 \end{pmatrix}, & [p_{i,i+1,2},\, q_{i,i+1,2}] &= [-1,\, -1],\\
\mathrm{a}_{i,i+1,3} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & k_{\mathrm{CO}_2}^{\mathrm{De}} & 0 \end{pmatrix}, & [p_{i,i+1,3},\, q_{i,i+1,3}] &= [-2,\, -1],\\
\mathrm{a}_{i,i+1,4} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & k_{\mathrm{CO}_2}^{\mathrm{De}} \\ 0 & 0 & 0 \end{pmatrix}, & [p_{i,i+1,4},\, q_{i,i+1,4}] &= [-1,\, -2],\\
\mathrm{a}_{i,i+1,5} &= \begin{pmatrix} 0 & 0 & 0 \\ k_{\mathrm{O}}^{\mathrm{Diff}} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, & [p_{i,i+1,5},\, q_{i,i+1,5}] &= [-1,\, +1],\\
\mathrm{a}_{i,i+1,6} &= \begin{pmatrix} 0 & k_{\mathrm{O}}^{\mathrm{Diff}} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, & [p_{i,i+1,6},\, q_{i,i+1,6}] &= [+1,\, -1],\\
\mathrm{a}_{i,i+1,7} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ k_{\mathrm{CO}}^{\mathrm{Diff}} & 0 & 0 \end{pmatrix}, & [p_{i,i+1,7},\, q_{i,i+1,7}] &= [-2,\, +2],\\
\mathrm{a}_{i,i+1,8} &= \begin{pmatrix} 0 & 0 & k_{\mathrm{CO}}^{\mathrm{Diff}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, & [p_{i,i+1,8},\, q_{i,i+1,8}] &= [+2,\, -2],
\end{aligned}
$$

for $i = 1, \ldots, d$, where $\mathrm{a}_{d,d+1,\mu} = \mathrm{a}_{d,1,\mu}$ and $[p_{d,d+1,\mu},\, q_{d,d+1,\mu}] = [p_{d,1,\mu},\, q_{d,1,\mu}]$. The output of Algorithm (16) is then an MME operator $\mathbf{A} \in \mathbb{R}^{(3\times3)\times\cdots\times(3\times3)}$ in TT format with ranks equal to 16, which is the same size as the operator in [6]. Using this exact tensor-train decomposition, we can compute stationary and time-dependent probability distributions by formulating eigenvalue problems or applying implicit time propagation schemes combined with ALS, see Chapter 4. In [6], we carried out several numerical experiments, which we repeat here with improved computational capabilities using the derived SLIM decomposition. The results of these computations are shown in the next section.

10.3. Numerical Experiments

10.3.1. Scaling with System Size

In this experiment, we examine the dependence of the computational complexity on the system size by increasing the number of sites and measuring the CPU time needed to approximate the stationary distribution for $p_{\mathrm{CO}} = 1\,\mathrm{atm}$ ($k_{\mathrm{CO}}^{\mathrm{Ad}} = 10^8 s^{-1}$). For each number of sites, we start with a fully O-covered surface represented by a tensor $\mathbf{P}_0$ with rank-one decomposition

$$\mathbf{P}_0 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Employing the implicit Euler method and using ALS to solve the resulting systems of linear equations given in (4.5.2), we start with a step size of 10^{-10} and double the step size after each step. For the first system of linear equations, we set the initial guess to a uniformly distributed tensor train, cf. Section 9.3.1, and use the solution of each iteration step as the initial guess for the next system. Different tests have shown that repeating ALS twice at each time step with a rank bound of 10 is sufficient to keep the residual errors given in (4.5.3) below 10^{-4} and to accurately approximate the stationary distribution within 20 steps, i.e. $\|\mathbf{A} \cdot \mathbf{P}_{20}\|_2$ is less than or close to 1 for all tested numbers of dimensions. For low dimensions, we observed that the relative error between the real stationary distribution and the vectorization of $\mathbf{P}_{20}$ is much smaller than 1%.

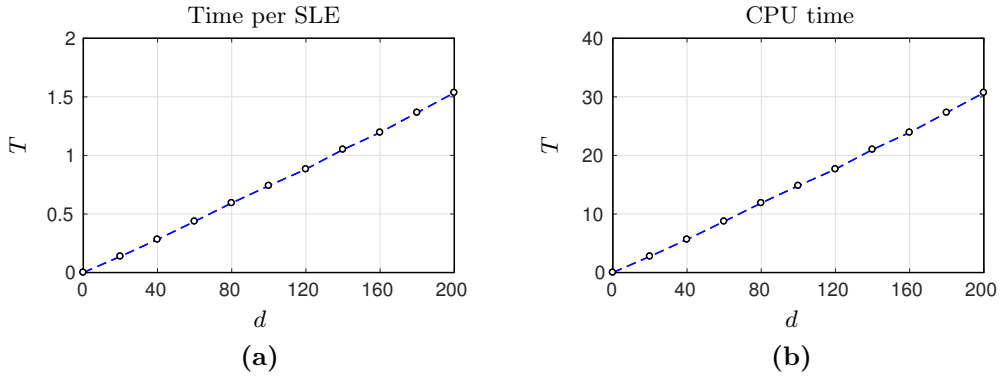


Figure 10.2: *CPU times for increasing number of dimensions: (a) CPU time in seconds over order d for computing the stationary distribution. (b) Average time in seconds over order d for solving one system of linear equations (SLE).*

Figure 10.2 shows the CPU times for the entire computations as well as the average time needed for solving a single system of linear equations in the TT format. It follows from the estimation of the computational complexity of ALS, see Table 4.1, that the CPU times scale linearly with the number of dimensions, cf. Figure 10.2 (a). This is also reflected in the time needed for solving the whole problem as shown in Figure 10.2 (b). Note that the largest systems of linear equations we consider have

$3^{200} \approx 10^{95}$ unknowns, which is far beyond the capabilities of classical methods for solving systems of linear equations. Using the TT approach, we are able to compute the numerical solution of the corresponding MME within approximately 30 seconds.

10.3.2. Varying the CO Pressure

In this section, we approximate the stationary distributions for several values of the CO pressure. We then extract key quantities describing the efficiency of the catalyst from these distributions, namely the *turn-over frequency* (TOF) and the *coverages*. The TOF represents the number of reactions of a specific type executed per unit time and unit surface. In our case, it measures how often CO_2 is produced, i.e.

$$\mathrm{TOF} = \frac{k_{\mathrm{CO_2}}^{\mathrm{De}}}{d} \sum_{i \in \{1,\ldots,d\}} \sum_{j \in \{i-1, i+1\}} \mathbb{P}(\mathrm{CO\ on\ } \Theta_i \wedge \mathrm{O\ on\ } \Theta_j), \tag{10.3.1}$$

where $\mathbb{P}(\mathrm{CO\ on\ } \Theta_i \wedge \mathrm{O\ on\ } \Theta_j)$ denotes the probability for finding CO on site Θ_i and O on a neighboring site Θ_j. Note that we here identify the sites Θ_j for $j = 0$ and $j = d + 1$ with Θ_d and Θ_1, respectively. Since the TOF depends strongly on the amount of CO molecules and oxygen atoms adsorbed on the catalytic surface, we also consider the coverages, which are the average numbers of the different occupation types per total number of sites, i.e. $\mathbb{P}(\varnothing \text{ on } \Theta_i)$, $\mathbb{P}(\mathrm{O\ on\ } \Theta_i)$, and $\mathbb{P}(\mathrm{CO\ on\ } \Theta_i)$ for any site Θ_i, $i = 1, \ldots, d$. Due to the homogeneous structure of the system, the coverages are translationally invariant, e.g. it holds that $\mathbb{P}(\varnothing \text{ on } \Theta_i) = \mathbb{P}(\varnothing \text{ on } \Theta_j)$ for different sites Θ_i and Θ_j. Additionally, the TOF given in (10.3.1) may be expressed without summation over all sites. Assuming the same reaction conditions as in [110], we vary the CO pressure between 10^{-4} atm and 10^2 atm and compare the results of the TT approach with results obtained by highly accurate kMC simulations carried out with the *kmos package* [51]. For both methods, we set the number of adsorption sites to 20.

Table 10.2 shows the used methods and TT ranks for different values of the CO pressure $p_{\mathrm{CO}} \in \left[10^{-4}, 10^2\right]$. For parameter values $p_{\mathrm{CO}} < 1\,\mathrm{atm}$, we obtain accurate approximations of the stationary distribution by applying ALS to an eigenvalue problem of the form (4.5.6). Systems with higher CO pressure are simulated by applying the implicit Euler method to the MME with an empty surface as initial state. We solve the systems of linear equations given in (4.5.2) by using ALS with the distribution computed in the previous step as input tensor. The initial guess for the system corresponding to the first iteration step is a uniformly distributed tensor train with rank bound as shown in Table 10.2. Note that all rank bounds for the TT approximations and also the number of steps and the step sizes of the implicit Euler method were determined through multiple numerical experiments. We implemented a simple step-size adaptation based on the alteration of $\|\mathbf{A} \cdot \mathbf{P}_k\|_2$ in each iteration step and tested different TT ranks for the first initial guess.

Table 10.2.: *Computation of stationary distributions for varying CO pressure: For the different CO pressures, either the stationary distribution is approximated by formulating an eigenvalue problem (EVP) or by employing the implicit Euler method (IEM). Additionally, we show the chosen bound of the TT ranks, the required computation time, and the closeness to the stationary distribution* $\|\mathbf{A} \cdot \mathbf{P}_{\text{stat}}\|_2$*, where* $\mathbf{A}$ *is the respective MME operator and* $\mathbf{P}_{\text{stat}}$ *is the computed approximation.*

p_{CO} in atm	Method	TT ranks	Closeness	CPU time
10^{-4}	EVP	4	0.003	3.44
$10^{-3.5}$	EVP	5	0.006	2.88
10^{-3}	EVP	6	0.004	8.36
$10^{-2.5}$	EVP	7	0.002	5.14
10^{-2}	EVP	11	0.008	3.52
$10^{-1.5}$	EVP	12	0.003	5.11
10^{-1}	EVP	12	0.005	4.25
$10^{-0.5}$	EVP	11	0.019	7.65
10^{0}	IEM	14	0.138	6.63
$10^{0.1}$	IEM	17	0.181	15.12
$10^{0.2}$	IEM	20	0.322	46.41
$10^{0.3}$	IEM	25	0.525	140.92
$10^{0.4}$	IEM	33	0.908	403.49
$10^{0.5}$	IEM	39	4.240	740.48
$10^{0.6}$	IEM	40	33.562	744.34
$10^{0.7}$	IEM	39	37.197	813.51
$10^{0.8}$	IEM	40	12.017	786.94
$10^{0.9}$	IEM	39	6.859	791.25
10^{1}	IEM	40	5.937	853.65
$10^{1.5}$	IEM	21	0.026	51.08
10^{2}	IEM	16	0.097	17.70

We only used TT ranks not higher than 40 with the aim to compute an approximation $\mathbf{P}_{\text{stat}}$ of the stationary distribution such that $\|\mathbf{A} \cdot \mathbf{P}_{\text{stat}}\|_2 < 1$. As one can see, this threshold is not reached for all CO pressures. From an algebraic point of view, the complexity of the corresponding stationary distributions in full tensor format cannot be accurately approximated with the same TT ranks as used for low CO pressures. An explanation for this effect is that the probability at low pressures is concentrated in a rather small number of possible surface arrangements, while the full tensors of the stationary distributions at high pressures are dense and the probabilities of a large amount of surface configurations do not vanish. Additionally, we observed that higher TT ranks do not necessarily imply a better accuracy, which can have multiple reasons such as the chosen scheme for the step size adaptation or the application of the ALS algorithm to non-symmetric systems. Overall, Table 10.2 shows that the TT ranks and CPU times increase as p_{CO} gets closer to $10^{0.7}$ and that it is easier to approximate the stationary distributions in terms of CPU time and closeness to the stationary distribution for low than it is for high CO pressures.

For the considered values of the CO pressure, we now want to investigate the correlation lengths of the cus sites. Since the system is translationally invariant, the *total correlation* [124] for the site distance $l \in \{1, \dots, 10\}$ is defined as the *Kullback–Leibler divergence* [125] from $\mathbb{P}(x_1 = i \wedge x_{1+l} = j)$ to $\mathbb{P}(x_1 = i)\mathbb{P}(x_{1+l} = j)$, i.e.

$$C(l) = \frac{1}{9} \sum_{i=1}^{3} \sum_{j=1}^{3} \mathbb{P}(x_1 = i \wedge x_{1+l} = j) \cdot \log \left(\frac{\mathbb{P}(x_1 = i \wedge x_{1+l} = j)}{\mathbb{P}(x_1 = i)\mathbb{P}(x_{1+l} = j)} \right),$$

where x_1 and x_{1+l} are the states of the active sites Θ_1 and Θ_{1+l}. Comparing Figure 10.3, where we show the normalized correlations $\tilde{C}(l) = C(l)/C(1)$ for different CO pressures, with Table 10.2, we see a close connection between the computational effort for approximating the stationary distribution and the correlation length for the corresponding CO pressure. The reason for this may be that much higher TT ranks would be needed in order to represent the dependence among the set of active sites. For high correlation lengths, the calculation of the TT approximations is more expensive than for low correlation lengths, where we are able to keep the TT ranks and CPU time at a small level.

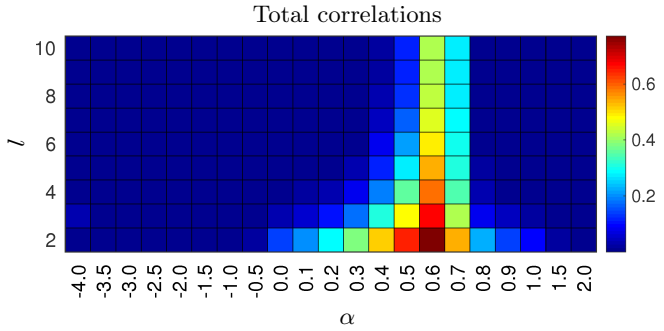


Figure 10.3: *Correlations of active sites for varying CO pressure: Values of the correlation function $\tilde{C}$ for distances $l = 2, \dots, 10$ over exponents α with $p_{CO} = 10^{\alpha}$ atm.*

Finally, Figure 10.4 shows the TOF over the CO pressure interval and an area plot of the coverages , respectively. In accordance with the results obtained by kMC simulations, we observe three characteristic regimes. At low CO pressures, the surface is almost fully O-covered, while at high pressures the surface is almost fully CO-covered. Additionally, there is an intermediate regime, where the probabilities for all possible site occupations do not vanish. Within this regime, the fraction of empty sites reaches its highest value of approximately 1.2%. The TOF is monotonically increasing on the CO pressure interval $\left[10^{-4}, 10^{0}\right]$ and has its maximum in the intermediate regime at $p_{\mathrm{CO}} \approx 5\,\mathrm{atm}$, were both reactants for the CO formation are available in sufficient amount on the surface. Additionally, the position of the peak corresponds to the longest correlation lengths as shown in Figure 10.3 and therefore to the most expensive calculations. We observe an almost perfect agreement when comparing the TOF obtained by the TT approach and by kMC simulations in Figure 10.4 (b). This shows that the presented approach is able to produce results with high accuracy while keeping the computational effort on a low level.

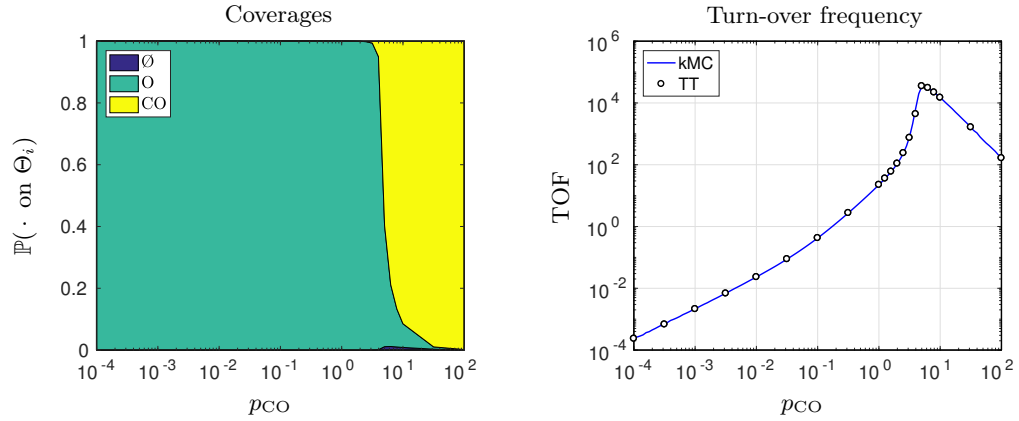


Figure 10.4: *Coverages and TOF for increasing CO pressure: (a) Mean coverages of the cus sites over CO pressure p_{CO}. Blue: empty, green: O-covered, yellow: CO-covered. (b) Turn-over frequency of the catalyst over p_{CO}. Blue line: TOF obtained with kMC, black circles: values computed with tensor-train approach*

10.3.3. Increasing the Oxygen Desorption Rate

We use the last experiment in this chapter to demonstrate the advantage of the TT approach over the kMC method when considering stiff problems, i.e. when the time scales of elementary reactions differ significantly from each other, cf. [126]. In terms of kMC simulations, the system fluctuates only between a small number of short-lived states and a high number of Monte Carlo steps is needed to ensure an sufficient sampling. If the stiffness of the system is increased even further, reaching a certain final time also requires a larger number of simulation steps.

Setting the CO pressure to $p_{\mathrm{CO}} = 10^{-4}$ atm, the catalytic surface (after relaxation time) is almost fully O-covered, see Figure 10.4 (a). The dominant processes then are the oxygen adsorption and desorption. We increase the stiffness of the system by multiplying the rate constant $k_{\mathrm{O_2}}^{\mathrm{De}}$ of the oxygen desorption by a parameter λ between 1 and 10^6. Considering 10 cus sites, we compute the numerical solution of the MME up to 1 second – starting with an empty surface and ending close to the steady state – by employing the kMC method and the tensor approach, respectively. For the TT approach, we use a combination of the implicit trapezoidal rule and the implicit Euler method to approximate the transient process. That is, starting with a step size $\tau = 10^{-11}$ and increasing the exponent of τ after each ten steps, we use the second-order trapezoidal rule for the first 40 iteration steps, where the probability distributions over the state space are changing rapidly. The implicit Euler method is used for the last 70 iteration steps since we observed a better convergence behavior for larger step sizes compared to the application of the trapezoidal rule. We suppose that this is a consequence of the implicit Euler method being L-stable, which means that instabilities are damped when using larger step sizes. Overall, we compute 110 steps with step sizes from 10^{-11} to 10^{-1}. The TT ranks are chosen in such a way that we keep the relative errors given in (4.5.3) and (4.5.5), respectively, below

0.05 and converge to an approximation $\mathbf{P}_{\text{stat}}$ of the stationary distribution with $\|\mathbf{A}\mathbf{P}_{\text{stat}}\|_2 \leq 1$.

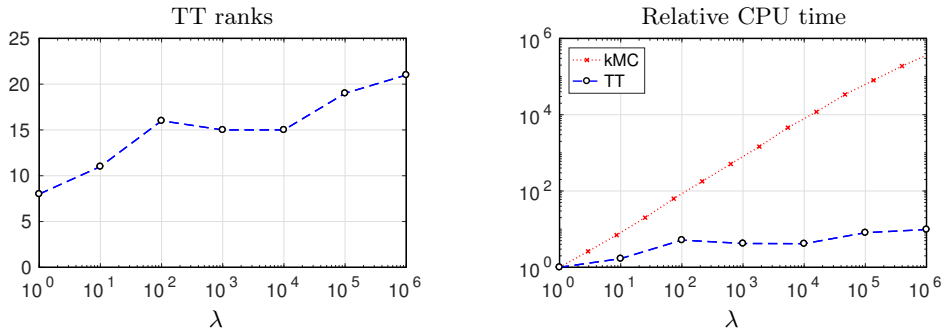


Figure 10.5: *Computational complexity for increasing oxygen desorption rate: (a) TT ranks used for computing the probability distributions. (b) Relative CPU times of the kMC and TT approach.*

Figure 10.5 shows the used TT ranks and the relative CPU times. We divided all CPU times by the time needed to compute the numerical solution for $\lambda = 1$ in order to compare the behavior of both approaches for systems with increasing stiffness. In absolute times, the kMC method is still faster than the TT approach. However, the relative CPU times for the kMC method increase linearly with λ while the computational complexity of the TT approach grows at much lower rate. The results indicate that the improvement of tensor-based (implicit) integration schemes may lead to a better performance than kMC simulations for problems with increasing stiffness. Considering Figure 10.6, we see that the norm of the computed probability distributions converges to approximately 1 for small values of λ. That is, the stationary distribution for these cases is concentrated in the state of a fully O-covered surface. For higher values of λ, the probability distributions spread over various surface configurations, which can be explained by the shorter times of oxygen atoms staying on the surface.

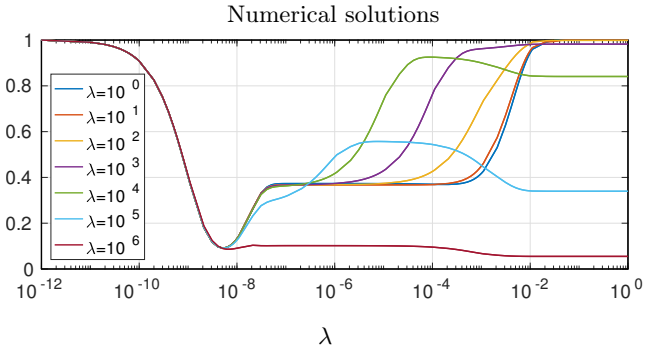


Figure 10.6: *Numerical solutions for increasing oxygen desorption rate: The 2-norms of the computed probability distributions are plotted for different values of the stiffness parameter. The norm of the approximated stationary distribution decreases for growing λ.*

11 Fluid Dynamics

In this chapter, we consider the application of TDMD to different simulations of fluid flows. After a brief description of the analysis of fluid dynamics problems by means of computer-based simulation, we show two illustrative examples. The first example is a time series showing wave patterns formed in the temperature field of a rotating fluid. The second example, which we also considered in [14], is a three-dimensional simulation of the flow around a blunt body. For both examples, we will apply the methods described in Section 7.3 in order to compute (exact) DMD modes and corresponding eigenvalues directly in the TT format.

11.1. Computational Fluid Dynamics

The term *computational fluid dynamics* (CFD) refers to the analysis of fluid flow problems arising from different subdisciplines of fluid dynamics such as aerodynamics [127, 128], hydrodynamics [129, 130], and heat transfer [131, 132]. The mathematical models of fluid flows are often given by *Navier–Stokes equations* and related fluid descriptions as the *Euler* or *Boussinesq equations*, see e.g. [133]. The general formulation of the Navier–Stokes equations for compressible flows, i.e. flows with non-constant fluid density, is

$$\rho\left(\frac{\partial\vec{v}}{\partial t}+(\vec{v}\cdot\nabla)\vec{v}\right)=-\nabla p+\mu\Delta\vec{v}+(\zeta+\mu)\nabla(\nabla\cdot\vec{v})+f, \tag{11.1.1}$$

where $\vec{v}$ is the fluid velocity, p is the pressure, ρ is the fluid density, and f is an external body force. The constants μ and ζ denote the dynamic viscosity (resistance to shearing flows) and the volume viscosity (resistance to volume compression and expansion), respectively. Together with the principles of mass and energy conservation, these equations constitute the basic mathematical model of CFD.

Depending on the underlying assumptions, the fluid velocity, its pressure and other quantities can be approximated on the computational domain by discretizing the model equations, e.g. by using the *finite volume method* [134] and solving the corresponding algebraic equations. For further details about CFD and fluid modeling, we refer to [133, 135].

11.2. Numerical Examples

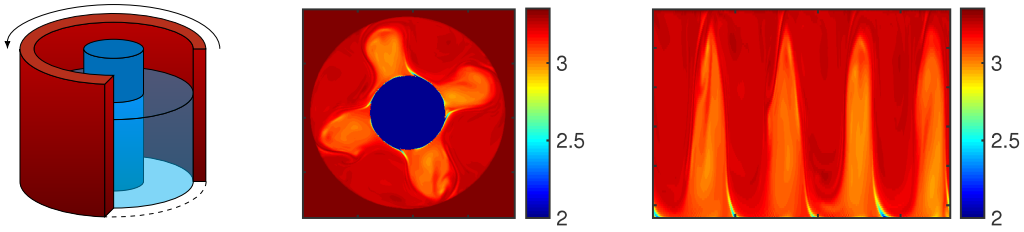
11.2.1. Rotating Annulus

The first example for the application of TDMD is a differentially heated annulus in rotation. Introduced by Fultz in 1959 as a laboratory model for atmospheric circulation [136], the underlying physics have been of great interest in experimental and numerical research, see e.g. [137, 138, 139, 140]. A schematic drawing of the apparatus is shown in Figure 11.1 (a). The rotating tank is filled with de-ionized water and the outer wall is heated, whereas the inner wall is cooled. If the rotation rate is high enough, drifting wave patterns in the temperature distribution are formed.

The system can be numerically modeled by a Boussinesq approximation. That is, we consider the incompressible case of the Navier–Stokes equations given in (11.1.1) together with an equation for the heat flow within the fluid. The three equations for the Boussinesq approximation are then given by

$$\begin{aligned} \frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\vec{v} &= -\frac{1}{\rho}\nabla p + \nu \Delta \vec{v} + \frac{\delta\rho}{\rho} g e_z - 2\Omega e_z \times \vec{v}, \\ \nabla \cdot \vec{v} &= 0, \\ \frac{\partial T}{\partial t} + (\vec{v} \cdot \nabla)T &= \kappa \Delta T. \end{aligned} \tag{11.2.1}$$

Here, $\nu = \mu/\rho$ denotes the kinematic viscosity of the fluid, $\delta\rho$ is the difference between the density of the given fluid parcel and the reference density ρ, and κ is the thermal diffusivity. The last two terms on the right-hand side of the first equation describe the gravitational and the Coriolis force, respectively, where e_z denotes the unit vector in vertical direction.



(a) **(b)** **(c)**

Figure 11.1: *Differentially heated rotating annulus: (a) Schematic drawing of the laboratory set-up. The annular gap ranges from 4.5 cm to 12 cm in radial direction and the water depth is 13.5 cm. The kinematic viscosity and the thermal diffusivity of the fluid are given by* $1.004\,mm^2/s$ *and* $0.1434\,mm^2/s$*, respectively. (b) A single representative snapshot of the temperature field (in terms of temperature difference) in the considered slice, viewed from above. (c) The same snapshot projected to a rectangle with periodic boundaries (at the left and right side). For both pictures, data values less than 2 K are mapped to the minimum of the colormap in order to compare both visualizations.*

Given time-series data computed with the EULAG flow solver [141], we want to compute the exact DMD modes of the flow from the temperature fields at different time points directly in the TT format, cf. (7.3.16). We assume a rotation rate of 0.635 rad/s and a radial temperature gradient of 6.7 K, i.e. the outer and the inner wall have a temperature of 16.65°C and 23.35°C, respectively. Under these forcing parameters, the formation of a four-fold symmetric wave pattern in the fluid can be observed. As an introductory example for the application of TDMD, we consider a single horizontal slice of the three-dimensional system. Similar patterns as shown in 11.1 (b) can be observed throughout all horizontal slices of the annulus. The fluid temperature fields of the different snapshots are then interpolated on a circular grid and projected to a rectangle with periodic boundaries, see Figure 11.1 (c). By restricting the analysis only to the essential part of the temperature field and projecting it to a rectangular grid, we obtain an almost linear relationship between time-shifted snapshots since, on a coarse level, the temperature distribution at time $t + \delta t$ is approximately the translated distribution at time t. The existence of a linear operator representing this relationship is the main assumption of DMD, see Section 7.3.

We extract one complete rotation of the wave pattern from the given data. That is, we consider 626 snapshots represented by matrices $\mathbf{T}_k \in \mathbb{R}^{106\times720}$, $k = 0, \dots, 625$. The time step between two consecutive snapshots is $\delta t = 0.954\,\text{s}$. As described in (7.3.3), we then define the tensors $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{106\times720\times625}$ as

$$\mathbf{X}_{:,:,k} = \mathbf{T}_{k-1} \quad \text{and} \quad \mathbf{Y}_{:,:,k} = \mathbf{T}_k, \tag{11.2.2}$$

for $k = 1, \dots, 625$.

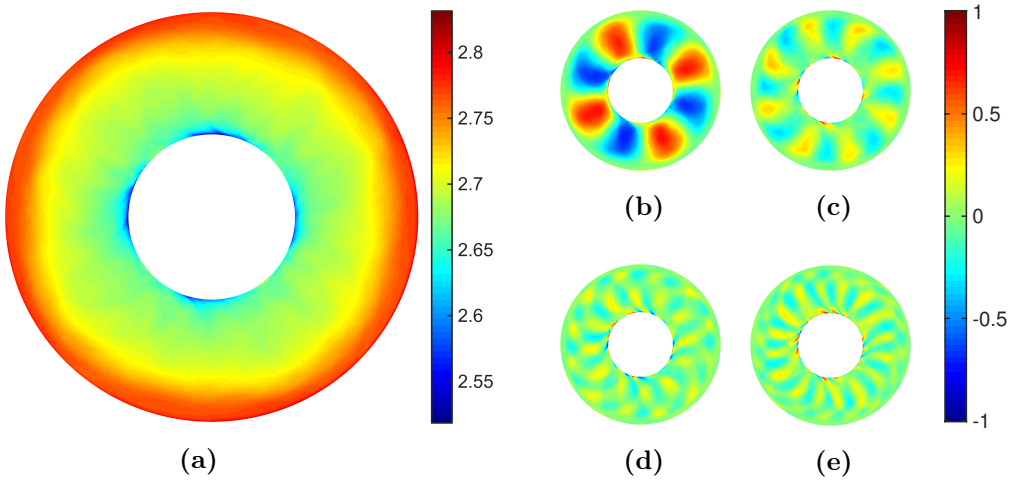


Figure 11.2: *Results for the rotating annulus: (a) DMD mode corresponding to eigenvalue $\lambda = 1$, which can be interpreted as the average temperature field. The other modes correspond to the following eigenvalues: (b) $\lambda \approx \exp(4 \cdot i \cdot (2\pi/625))$. (c) $\lambda \approx \exp(8 \cdot i \cdot (2\pi/625))$. (d) $\lambda \approx \exp(12 \cdot i \cdot (2\pi/625))$. (e) $\lambda \approx \exp(16 \cdot i \cdot (2\pi/625))$. Only the real parts of the DMD modes are shown.*

Table 11.1.: *TDMD applied to the rotating annulus: Influence of the truncation parameter ε on the ranks, runtimes, and accuracy of the first two considered DMD modes.*

			Mode (a)		Mode (b)	
Threshold	TT ranks	CPU time	e_λ	e_φ	e_λ	e_φ
$\varepsilon = 0$	[1, 106, 625 , 1]	2.85 s	0	0	0	0
$\varepsilon = 1$	[1, 61, 625 , 1]	2.69 s	1.85e−10	6.12e−06	9.46e−09	3.56e−04
$\varepsilon = 5$	[1, 38, 316 , 1]	0.67 s	2.41e−07	3.17e−04	6.71e−06	6.49e−03
$\varepsilon = 10$	[1, 30, 169 , 1]	0.29 s	3.23e−07	8.89e−04	6.82e−05	2.88e−02

Without truncation, the TT ranks of both tensors are given by $[1, 106, 625, 1]$. As we have discussed in [14], we assume that the data matrices $\mathbf{X}$ and $\mathbf{Y}$ are already given in the TT format. For instance, that could mean that the set of partial differential equations of the form (11.2.1) is numerically solved by applying an appropriate time-stepping scheme combined with (M)ALS. However, we here apply Algorithm 2 to convert the snapshot tensors given in (11.2.2) into the TT format. Thus, we can reduce the TT ranks and the computation time for exact TDMD by increasing the threshold ε in Algorithm 2. We considered another two-dimensional flow problem in [14], where we showed that one can significantly decrease the computation time for TDMD while obtaining only small numerical errors for the low-rank approximation. The same applies for the rotating annulus. For $\varepsilon = 0$, the computation time for the TT approach (the runtime of Algorithm 2 not included) is approximately 3 s, whereas the CPU time for the classical DMD method using matricizations of $\mathbf{X}$ and $\mathbf{Y}$ is around 6 s. For $\epsilon > 0$, the computation times are even much smaller than 3 s. Five of the computed DMD modes can be seen in Figure 11.2 and Table 11.1 shows – corresponding to the different values of ε – the TT ranks, CPU times, and approximation errors e_λ and e_φ defined as

$$e_\lambda = \frac{\left|\lambda - \tilde{\lambda}\right|}{|\lambda|} \qquad \text{and} \qquad e_\varphi = \frac{\|\varphi - \tilde{\varphi}\|_2}{\|\varphi\|_2}, \tag{11.2.3}$$

respectively, where λ and φ are the DMD eigenvalue and mode for $\varepsilon = 0$ and $\tilde{\lambda}$ and $\tilde{\varphi}$ are the approximations for $\varepsilon > 0$. We normalized each mode such that the largest absolute value is 1. Almost all computed eigenvalues lie on the unit circle, i.e. most eigenvalues λ can be written as $\lambda = \exp(i \cdot k \cdot (2\pi/625))$, where i is the imaginary unit and $k \in \mathbb{N}$. For each mode φ, the corresponding frequency, see e.g. [142, 143], is then defined as

$$\omega = \frac{\Im(\log(\lambda))}{\Delta t} \approx \frac{k \cdot 2\pi}{t^*}, \tag{11.2.4}$$

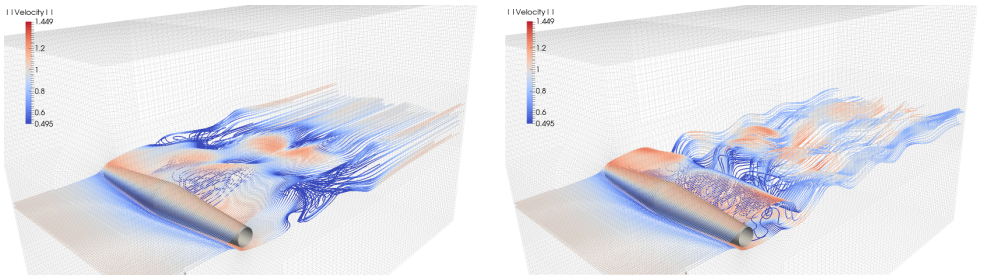
where λ is the eigenvalue corresponding to φ and t^* is the total time for one complete rotation of the wave pattern. One can see in Figure 11.2 that larger structures correspond to lower frequencies, whereas smaller structures correspond to higher frequencies, cf. [57]. Note that these frequencies are also reflected in the patterns shown in Figure 11.2 (b)–(e).

11.2.2. Flow Around a Blunt Body

The second example, which we already considered in [14], shows the flow around a blunt body in three dimensions on the domain $\Omega = [0, 25] \times [0, 15] \times [0, 10]$. Inside the domain, we place a conical object with the center axis at $(x_1, x_2) = (5, 7.5)$ and diameters given by $D_1 = 0.8$ at the boundaries and $D_2 = 1.6$ in the middle of the channel. See Figure 11.3, where we display the streamlines of the velocity field at two different time steps. Additionally, periodic boundary conditions in the x_2- and the x_3-direction are applied. The mathematical model of this system is given by the incompressible Navier–Stokes equations, i.e. the velocity field $\vec{v}$ is divergence-free and (11.1.1) simplifies to

$$\rho \left(\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\vec{v} \right) = -\nabla p + \mu \Delta \vec{v} + f.$$

Similar examples governed by solving the three-dimensional incompressible Navier–Stokes equations can be found in [142, 144, 145].



(a) **(b)**

Figure 11.3: *Simulation of the flow around a blunt body: The flow is visualized at different time points $t = 30\,s$ and $t = 300\,s$, respectively, by using streamlines which are inserted at the inflow and slightly above and below the cone axis, respectively. The streamlines are colored corresponding to the velocity magnitude.*

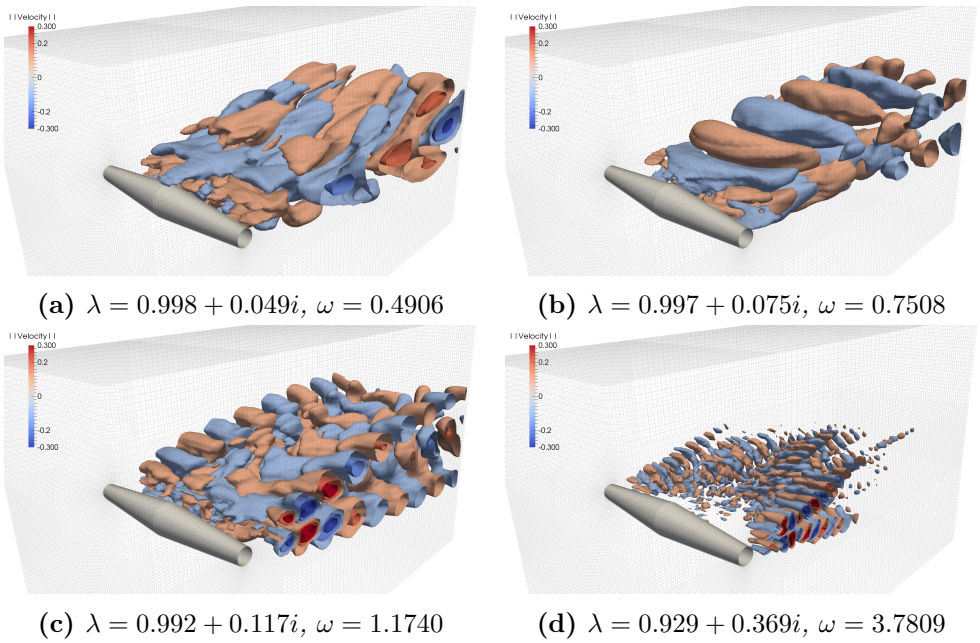
Using the OpenFOAM toolbox [146] and discretizing the domain by a rectangular grid with approximately 10^6 degrees of freedom, we compute 1001 snapshots of the system. These snapshots are then interpolated on an equidistant, rectangular grid, where $n_1 = 150$, $n_2 = 85$, and $n_3 = 80$ are the numbers of grid points in each dimension. Here, we are interested in the velocity magnitude $|\vec{v}| = \sqrt{v_1^2 + v_2^2 + v_3^2}$. That is, the tensors $\mathbf{X}$ and $\mathbf{Y}$, which contain the (shifted) snapshots, are elements of the tensor space $\mathbb{R}^{150 \times 85 \times 80 \times 1000}$. Figure 11.4 shows a few computed DMD modes with their corresponding frequency, which is given by (11.2.4). Similar to the previous example, low frequencies indicate slowly rotating vortices, whereas high frequencies indicate fast rotating vortices. Figure 11.4 also shows that larger structures again correspond to lower frequencies and smaller structures correspond to higher frequencies. Additionally, the results reveal a tendency for larger structures to originate from the middle of the conical object and for smaller structures from the (periodic) boundaries of the object.

Table 11.2.: *TDMD applied to the blunt body problem: Influence of the truncation parameter ε on the TT ranks, runtimes, and accuracies of the leading DMD modes.*

			Mode (a)		Mode (b)	
Threshold	TT ranks	CPU time	e_λ	e_φ	e_λ	e_φ
$\varepsilon = 0$	[1, 150, 6083, 1000, 1]	134 s	0	0	0	0
$\varepsilon = 0.01$	[1, 150, 4708, 1000, 1]	102 s	6.29e−07	6.33e−04	1.53e−07	2.88e−04
$\varepsilon = 0.05$	[1, 150, 3649, 641, 1]	52 s	1.11e−04	3.64e−02	3.05e−05	2.37e−02
$\varepsilon = 0.1$	[1, 148, 3003, 527, 1]	35 s	1.38e−04	6.92e−02	1.26e−04	3.65e−02
$\varepsilon = 0.5$	[1, 135, 1624, 343, 1]	14 s	2.62e−04	8.69e−02	9.61e−05	5.55e−02
$\varepsilon = 1$	[1, 130, 1199, 278, 1]	8 s	4.34e−04	1.72e−01	1.80e−04	1.06e−01

Note that storing the tensor operator $\mathbf{A}$ (7.3.2) as well as its matricization A (7.3.4) in full format would require more than 7.5 TB. By applying Algorithm 2, we convert the tensors $\mathbf{X}$ and $\mathbf{Y}$ into the TT format using different thresholds ε in order to illustrate the efficiency of TDMD particularly for low TT ranks. In Table 11.2, where we compare the approximation of the first two modes, the resulting ranks, runtimes, and relative errors (11.2.3) for increasing values of ε are shown. As one can see, the initially high ranks of $\mathbf{X}$ and $\mathbf{Y}$ can be reduced without losing too much accuracy in the approximations of the DMD modes and corresponding eigenvalues. For $\varepsilon = 0$, the runtime of the TDMD approach is slightly higher than the runtime of conventional DMD for this problem, which is approximately 125 s. However, the time can be significantly decreased using different thresholds in $(0, 1]$.

All pictures shown in this section and further visualizations can also be found in [14].



(a) $\lambda = 0.998 + 0.049i$, $\omega = 0.4906$ **(b)** $\lambda = 0.997 + 0.075i$, $\omega = 0.7508$

(c) $\lambda = 0.992 + 0.117i$, $\omega = 1.1740$ **(d)** $\lambda = 0.929 + 0.369i$, $\omega = 3.7809$

Figure 11.4: *Results for the flow around a blunt body: Modes with corresponding eigenvalues close to 1 are visualized by iso-surfaces of the velocity magnitude.*

12
Brownian Dynamics

The aim of this chapter is to gain insight into the global behavior of Brownian systems by analyzing the eigenvalues and eigenfunctions of the corresponding Perron–Frobenius operator with the aid of Ulam's method, see Chapter 8. For this purpose, we will consider two examples for the approximation of the Perron–Frobenius operator in the TT format. We will use the Algorithms 13 and 14 to approximate the eigenfunctions and therefore to find invariant densities and metastable conformations of the considered systems. Note that this chapter only shows the first steps towards the approximation of transfer operators and their eigenfunctions in the TT format using (M)ALS for eigenvalue problems. There are still open questions such as how to reduce the ranks of the computed TT operator without overly changing the approximated dynamics or how to find the optimal orientation of coordinate axes in order to represent the eigenfunctions by low-rank tensors, cf. [13].

12.1. Langevin Equation

Brownian dynamics denote methods to simulate the irregular motion of particles in a fluid due to collisions with the solvent molecules. The standard mathematical model for the dynamics of these molecular systems is the *Langevin equation* [147]. The general form of this *stochastic differential equation* (SDE), see [148], is given by

$$m\frac{d^2X}{dt^2}(t) = -\nabla V(X(t)) - \gamma\frac{dx}{dt}(t) + R(t),$$

where m is the mass of the particle, V is an external potential, γ is a damping coefficient, and R is a stochastic process representing the effect of the collisions with the molecules of the fluid. Assuming the presence of an external force field, Brownian dynamics can be described by an *overdamped Langevin equation*. That is, because of the small mass of the considered particles, it is assumed that the inertia of these particles is negligible compared to the damping force. An overdamped Langevin equation can be written in differential form as

$$dX(t) = -\nabla V(X(t))dt + \sigma dW(t), \tag{12.1.1}$$

where the parameter σ is called diffusion coefficient. The force acting on a particle at various positions in space is given by the conservative force field $F = -\nabla V$ and the Brownian motion is realized by the (multidimensional) *Wiener process* W, see e.g. [48].

In the next section, we will consider two simple examples for overdamped Langevin equations in order to illustrate the efficiency of the TT approach when approximating eigenfunctions of the Perron–Frobenius operator and identifying metastable conformations.

12.2. Numerical Experiments

12.2.1. Two-Dimensional Triple-Well Potential

In order to compare the results obtained by the TT approach with the exact eigenpairs of an approximation of the Perron–Frobenius operator, we first choose a two-dimensional example on a rather small domain. We consider the Brownian motion of particles in a potential V taken from [60], which is defined as

$$\begin{aligned} V(x,y) = 3\,e^{-x^2-(y-\frac{1}{3})^2} - 3\,e^{-x^2-(y-\frac{5}{3})^2} - 5\,e^{-(x-1)^2-y^2} \\ - 5\,e^{-(x+1)^2-y^2} + \tfrac{2}{10}\,x^4 + \tfrac{2}{10}\left(y-\tfrac{1}{3}\right)^4, \end{aligned}$$

where $(x,y) \in \mathcal{S} = [-2,2] \times [-1,2]$ is the position of a particle. See Figure 12.1 for a visualization of the potential.

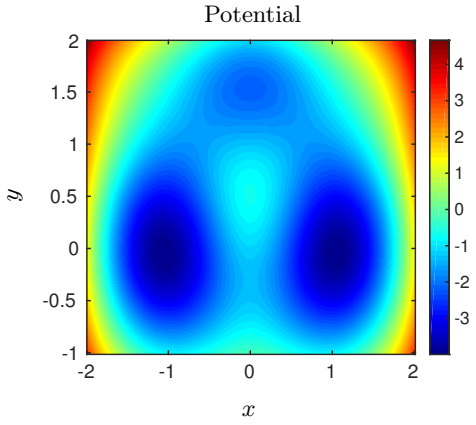


Figure 12.1: *Two-dimensional triple-well potential: Two deep wells are located at* $(-1,0)$ *and* $(1,0)$*, respectively, and a shallow well is located at* $(0,1.5)$.

The corresponding SDE in differential form is given by (12.1.1) with $X = (x,y)$ and $\sigma = 1.09$. The domain $\mathcal{S}$ is subdivided into 50×50 boxes of equal size and the *Euler–Maruyama method* [149] is used to compute the trajectories of the randomly generated test points. We set the step size to $h = 10^{-5}$ and integrate the SDE with initial conditions given by the positions of the test points from $t_0 = 0$ to $t_1 = 0.1$.

In order to apply Algorithm 7, we recorded the transitions of 500 test points per box. The corresponding approximation of the Perron–Frobenius operator is then a TT operator $\mathbf{P} \in \mathbb{R}^{(50\times50)\times(50\times50)}$ with ranks $[1, 1459, 1]$. That is, all entries of $\mathbf{P}$ can be represented by a TT decomposition with a rank nearly half as high as the rank of a full-rank TT operator with $[1, 2500, 1]$. However, as we already mentioned in Section 8.2, the storage consumption of $\mathbf{P}$ in the TT format is slightly larger than the storage consumption of the matricization of $\mathbf{P}$ in sparse format.

After computing the approximation of the Perron–Frobenius operator in the TT format by using Algorithm 7, we now want to analyze the slow dynamics of the system represented by the left-eigentensors of $\mathbf{P}$. In contrast to [13], where a global power iteration method was applied to the system, we here approximate the three leading eigenvalues $\lambda_1, \lambda_2, \lambda_3$ and corresponding eigentensors $\varphi_1, \varphi_2, \varphi_3 \in \mathbb{R}^{50\times50}$ of $\mathbf{P}$ by using ALS in combination with the BTT format, see Sections 4.3 and 3.5.2, respectively. The results are shown in Figure 12.2. As already discussed in [62], the system has different metastable sets located at the wells of the potential. The second eigenfunction depicted in Figure 12.2 (b) separates the two deep wells and is close to zero at the shallow well, whereas the third eigenfunction separates the two deep wells from the shallow well, see Figure 12.2 (c). In Figure 12.2 (d) and (e), respectively, we also show the second and the third eigenfunctions of the Koopman operator, which are represented by the right-eigentensors of $\mathbf{P}$ and correspond to the same eigenvalues λ_2 and λ_3, respectively. As described in Section 8.1, these eigenfunctions encode the same information as the eigenfunctions of the Perron–Frobenius operator.

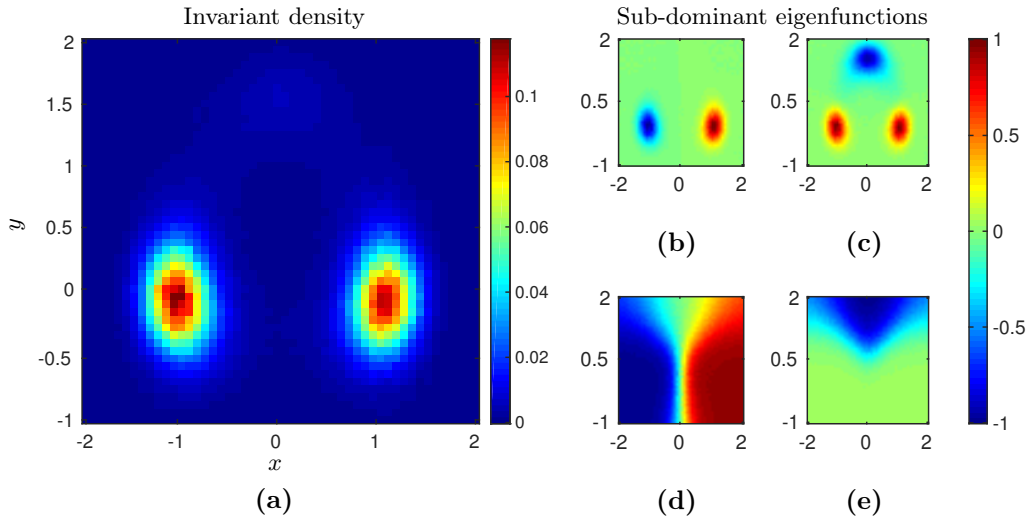


Figure 12.2: *Results for the triple-well potential: (a) Eigentensor of the Perron–Frobenius operator corresponding to eigenvalue $\lambda_1 = 1$ representing the invariant density of the system. (b) Eigentensor corresponding to $\lambda_2 = 0.9919$. (c) Eigentensor corresponding to $\lambda_3 = 0.9249$. The approximated eigenfunctions of the Koopman operator corresponding to λ_2 and λ_3, respectively, are shown in (d) and (e).*

Table 12.1.: *Approximation of the Perron–Frobenius operator for the triple-well potential: Eigenvalues and corresponding approximation errors.*

k	λ_k	e_{λ_k}	e_{φ_k}
1	1.0000	4.61e−04	3.83e−02
2	0.9919	5.04e−07	3.33e−02
3	0.9249	6.61e−05	3.48e−02

Note that the TT operator $\mathbf{P}$ is not symmetric, which is the main assumption for applying ALS to eigenvalue problems. However, Table 12.1 shows that we still are able to compute accurate approximations of the eigenpairs (λ_k, φ_k), $k = 1, 2, 3$, by a direct application of the algorithm. We set the TT ranks of the initial guess to $[1, 11, 1]$, which is the smallest possible TT rank such that the relative errors $e_{\varphi_k} = \|\varphi_k - \tilde{\varphi}_k\|_2 / \|\varphi_k\|_2$ are below 10% after two runs of ALS. Here, φ_k is the exact eigenvector computed by using classical methods and $\tilde{\varphi}_k$ is the vectorization of the eigentensor obtained from the TT approach. Furthermore, the relative errors of the corresponding eigenvalues given by $e_{\lambda_k} = |\lambda_k - \tilde{\lambda}_k| / |\lambda_k|$ are even smaller, which shows the applicability of ALS to non-symmetric eigenvalue problems.

12.2.2. Three-Dimensional Quadruple-Well Potential

As the second example, we consider the Brownian motion of particles in a three-dimensional potential V given by

$$V(x, y, z) = V_x(x) + V_y(y) + V_z(z) = (x^2 - 1)^2 + (y^2 - 1)^2 + z^2,$$

with $(x, y, z) \in \mathcal{S} = [-2, 2] \times [-2, 2] \times [-2, 2]$. In Figure 12.3, we plot the components V_x, V_y, and V_z for the respective domains.

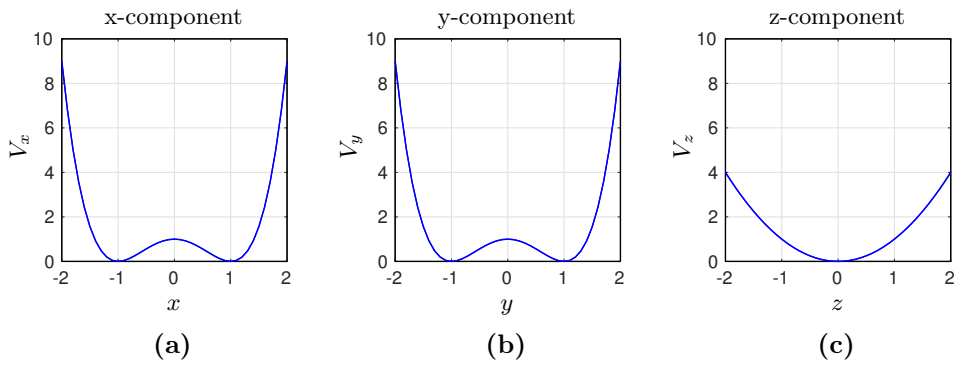


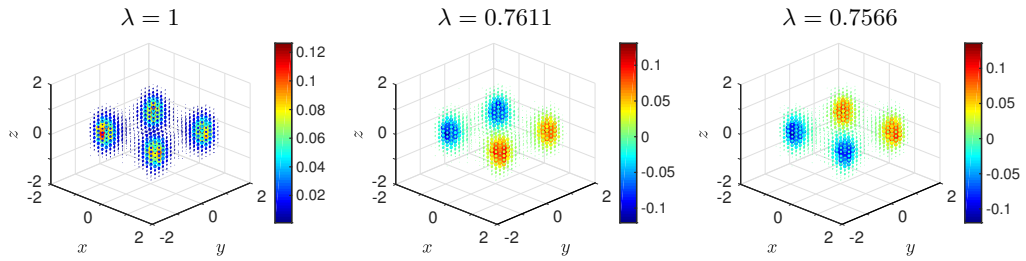
Figure 12.3: *Three-dimensional quadruple-well potential: The respective components in x- and y-direction are biquadratic functions. The z-component of the potential is a parabola.*

Table 12.2.: *Approximation of the dominant eigenpairs for the quadruple-well potential with threshold $\varepsilon = 0$: Eigenvalues and corresponding approximation errors.*

k	λ_k	e_{λ_k}	e_{φ_k}
1	1.0000	2.66e−15	2.93e−14
2	0.7611	2.92e−15	1.29e−13
3	0.7566	1.91e−15	2.60e−13

The corresponding Langevin equation is given by (12.1.1) with $X = (x, y, z)$ and $\sigma = 0.7$. This time, we subdivide the domain $\mathcal{S}$ into $25 \times 25 \times 25$ boxes and simulate 100 test points per box, whose trajectories are computed by the Euler–Maruyama method with step size $h = 10^{-3}$ and a lag time of 10. That is, the TT operator $\mathbf{P}$ is constructed by using Algorithm 8 with a list of 1562500 transitions as input. Even though the potential can be written as the sum of one-dimensional functions, the numerically computed approximation of the Perron–Frobenius operator cannot be represented as a low-rank TT operator. The TT ranks of $\mathbf{P}$, which are given by $[1, 509, 493, 1]$, are nearly as large as the ranks of a full-rank TT operator given by $[1, 25^2, 25^2, 1]$. Note that the TT operator $\mathbf{P}$ is again not symmetric.

In order to compute the eigentensors of $\mathbf{P}$, we apply the MALS algorithm for eigenvalue problems in combination with the BTT format, see Appendix A.2.5. Here, we do not restrict the TT ranks of the eigentensors, but apply truncated SVDs for the splitting of the computed super-cores, i.e. we discard all singular values σ_i with $\sigma_i/\sigma_1 < \varepsilon$, where ε is a given threshold. As before, we assume that the singular values are sorted in decreasing order. For $\varepsilon = 0$, we compute the three dominant eigenvalues and corresponding eigentensors, whose TT ranks are then given by $[1, 75, 25, 1]$. That is, MALS constructs an exact full-rank BTT representation of the three eigentensors. Table 12.2 shows the computed eigenvalues as well as the relative errors e_{λ_k} and e_{φ_k} as defined in the previous section. The eigentensors of the TT approximation of the Perron–Frobenius operator are shown in Figure 12.4.



(a) **(b)** **(c)**

Figure 12.4: *Results for the quadruple-well potential: The eigentensors corresponding to the dominant eigenvalues are shown in a three-dimensional scatter plot, where the color and size of the circles is determined by the entries of the eigentensors.*

Table 12.3.: *Approximation of the dominant eigenpairs for the quadruple-well potential with thresholds $\varepsilon > 0$: Influence of the truncation parameter ε on the ranks, runtimes, and accuracy of the dominant eigenpair.*

Threshold	BTT ranks	CPU time	e_{λ_1}	e_{φ_1}
$\varepsilon = 10^{-5}$	$[1, 62, 21, 1]$	11.2484 s	6.44e−15	8.57e−06
$\varepsilon = 10^{-4}$	$[1, 59, 20, 1]$	9.7492 s	7.48e−07	4.02e−05
$\varepsilon = 10^{-3}$	$[1, 56, 18, 1]$	8.3734 s	2.65e−05	1.12e−03
$\varepsilon = 10^{-2}$	$[1, 26, 10, 1]$	3.9754 s	4.83e−04	2.81e−02
$\varepsilon = 10^{-1}$	$[1, \ 2, \ 1, 1]$	2.6695 s	2.09e−04	9.83e−02
$\varepsilon = 10^{0}$	$[1, \ 1, \ 1, 1]$	2.3911 s	5.71e−03	3.17e−01

As expected, the system has four metastable sets at the wells of the potential. Furthermore, these wells are separated pairwise by the eigenfunctions φ_2 and φ_3. The CPU time for the computation of the eigenpairs using MALS with $\varepsilon = 0$ is approximately 15.5 s. We now increase the threshold ε for the truncated SVDs and compare the result for the dominant eigenpair with the exact solutions obtained by applying classical methods (CPU time of approximately 8.4 s). As we see in Table 12.3, we can reduce the computation time by truncating the ranks of the eigentensors given in BTT format and still obtain very accurate approximations of λ_1 and φ_1 for most of the considered truncation parameters. In particular, it is possible to approximate the eigentensor corresponding to the eigenvalue $\lambda = 1$ with TT ranks $[1, 2, 1, 1]$ and a relative error of less than 10%. Relying on future improvements of the proposed methods, this example shows that the TT approach for the approximation of eigenfunctions of the Perron–Frobenius and other transfer operators can be a powerful tool for certain problems.

13
Summary and Conclusion

In this thesis, we demonstrated the broad spectrum of application areas where tensor-based approaches are promising tools for tackling high-dimensional problems. In particular, we investigated the benefit of the TT format for the numerical solution of master equations in tensor notation and for the approximation of eigenvalues and eigentensors. We provided exact TT decompositions of linear tensor operators and described how to compute stationary and time dependent distributions. The overall aim was to mitigate the curse of dimensionality by reducing the memory consumption and the computational costs of various numerical problems. We showed that, at the same time, the TT format can provide accurate results even when restricting/truncating the TT ranks of our approximations. Using the TT format and its modifications, we were able to gain insight into the dynamics and structures of different problems. Some of these problems could not be solved before using classical numerical methods. The main contributions of this thesis were:

- tensor-based representations of Markovian master equations,
- tensor decompositions of linear operators representing systems based on nearest-neighbor interactions,
- tensor-based extension of the dynamic mode decomposition,
- TT approximations of Perron–Frobenius operators and their eigenvalues and corresponding eigenfunctions

After we have shown how to reformulate a given master equation in tensor notation, we introduced the SLIM decomposition for nearest-neighbor interaction systems and presented algorithms which can be used to automatically construct this decomposition for Markovian generators. By exploiting the coupling structure of such systems, we successfully demonstrated that it is possible to compute low-rank tensor decompositions of high-dimensional tensors representing time-dependent and stationary probability distributions, respectively. In particular, the storage consumption of SLIM decompositions increases only linearly with the network size for homogeneous systems. We expect this specific TT decomposition to be suitable for many systems based on nearest-neighbor interactions. Here, we considered examples for chemical reaction networks and the CO oxidation on a catalytic surface but also other nearest-neighbor interaction systems are of high interest for us. Additionally, we want to generalize our proposed method to more complex processes such as

next-nearest-neighbor interaction systems and other systems with certain coupling structures. An open question in this context is the optimal arrangement of the cores of a tensor network since the concept of nearest-neighbor interactions showed that the TT format is extremely sensitive to the correlation between the dimensions of the state space.

For the tensor-based dynamic mode decomposition, we showed that the TT cores implicitly contain information about the pseudoinverse of certain tensor unfoldings. Under the assumption that the data is already given in the TT format, we can therefore efficiently compute DMD modes directly as low-rank tensor representations. We demonstrated the performance of TDMD for two different fluid dynamics problems. Compared to the classical DMD algorithm, we were able to reduce the time for the computation of the modes significantly. Other variants and extensions of DMD related to different transfer operators such as the Koopman operator will be included in our future work.

In order to compute finite-dimensional approximations of the Perron–Frobenius operator for dynamical systems, we explained how to exploit the TT format and presented corresponding algorithms for two- and three-dimensional systems. We tested the tensor-based approach on two examples for the Brownian motion of particles under the influence of force fields. Our experiments showed that it is possible to compute accurate low-rank approximations of several eigenfunctions of the Perron–Frobenius operator simultaneously. However, as a first step towards the TT approximation of transfer operators, we assumed that the force fields were given by well-aligned potentials. By finding the optimal orientation of the coordinate axes for more complex dynamics, it might be possible to represent weak couplings between different dimensions by TT cores with small ranks. Our future research will include the approximation of other transfer operators and their eigenfunctions as well as higher-dimensional problems, e.g. molecular systems with a larger number of degrees of freedom, in order to analyze the scaling and to improve the efficiency of the proposed methods.

To solve optimization problems in the form of systems of linear equations and eigenvalue problems, respectively, we employed the ALS and MALS algorithms. In general, the convergence of these methods is still an open issue. Local convergence has been proven, but only under conditions that are hard to verify. An important aspect is that both algorithms are particularly designed for systems with symmetric (and positive definite) TT operators. Nevertheless, we obtained highly accurate approximations of the solutions for various non-symmetric problems. This is an effect which cannot be entirely explained at the moment. The development of tensor-based solvers for non-symmetric operators as well as suitable preconditioners may be a major topic in the future. Additionally, the combination with higher-order time integration schemes and the application to different ordinary and partial differential equations such as the Fokker–Planck equation is planned.

In our opinion, the further improvement of tensor-based approaches is a promising direction for mitigating the curse of dimensionality for many high-dimensional problems. We believe that the mathematical framework presented in this work is another step towards a broader applicability of tensor decompositions.

14 References

[1] S. R. White, Density matrix formulation for quantum renormalization groups, Physical Review Letters 69 (19) (1992) 2863–2866. doi:10.1103/PhysRevLett.69.2863.

[2] H. D. Meyer, F. Gatti, G. A. W. (eds.), Multidimensional quantum dynamics: MCTDH theory and applications, Wiley-VCH Verlag GmbH & Co. KGaA, 2009. doi:10.1002/9783527627400.ch3.

[3] T. Jahnke, W. Huisinga, A dynamical low-rank approach to the chemical master equation, Bulletin of Mathematical Biology 70 (8) (2008) 2283–2302. doi:10.1007/s11538-008-9346-x.

[4] V. Kazeev, M. Khammash, M. Nip, C. Schwab, Direct solution of the chemical master equation using quantized tensor trains, PLoS Comput Biol 10 (3) (2014) e1003359. doi:10.1371/journal.pcbi.1003359.

[5] S. Dolgov, B. Khoromskij, Simultaneous state-time approximation of the chemical master equation using tensor product formats, Numerical Linear Algebra with Applications 22 (2) (2015) 197–219. doi:10.1002/nla.1942.

[6] P. Gelß, S. Matera, C. Schütte, Solving the master equation without kinetic Monte Carlo: Tensor train approximations for a CO oxidation model, Journal of Computational Physics 314 (2016) 489–502. doi:10.1016/j.jcp.2016.03.025.

[7] P. Buchholz, Product form approximations for communicating Markov processes, Performance Evaluation 67 (9) (2010) 797–815. doi:10.1016/j.peva.2009.12.005.

[8] D. Kressner, F. Macedo, Low-rank tensor methods for communicating Markov processes, Quantitative Evaluation of Systems, Lecture Notes in Computer Science 8657 (2014) 25–40.

[9] P. Gelß, S. Klus, S. Matera, C. Schütte, Nearest-neighbor interaction systems in the tensor-train format, Journal of Computational Physics 341 (2017) 140–162. doi:10.1016/j.jcp.2017.04.007.

[10] G. Beylkin, J. Garcke, M. J. Mohlenkamp, Multivariate regression and machine learning with sums of separable functions, SIAM Journal on Scientific Computing 31 (3) (2009) 1840–1857. doi:10.1137/070710524.

[11] A. Novikov, D. Podoprikhin, A. Osokin, D. Vetrov, Tensorizing neural networks, in: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems 28 (NIPS), Curran Associates, Inc., 2015, pp. 442–450. arXiv:1509.06569v2.

[12] N. Cohen, O. Sharir, A. Shashua, On the expressive power of deep learning: A tensor analysis (2015). arXiv:1509.05009.

[13] S. Klus, C. Schütte, Towards tensor-based methods for the numerical approximation of the Perron–Frobenius and Koopman operator, Journal of Computational Dynamics 3 (2). doi:10.3934/jcd.2016007.

[14] S. Klus, P. Gelß, S. Peitz, C. Schütte, Tensor-based dynamic mode decomposition, ArXiv e-prints: arXiv:1606.06625.

[15] W. Hackbusch, S. Kühn, A new scheme for the tensor representation, The journal of Fourier analysis and applications 15 (5) (2009) 706–722. doi:10.1007/s00041-009-9094-9.

[16] I. V. Oseledets, A new tensor decomposition, Doklady Mathematics 80 (1) (2009) 495–496. doi:10.1134/S1064562409040115.

[17] I. V. Oseledets, E. E. Tyrtyshnikov, Breaking the curse of dimensionality, or how to use SVD in many dimensions, SIAM Journal on Scientific Computing 31 (5) (2009) 3744–3759. doi:10.1137/090748330.

[18] J. Ballani, L. Grasedyck, A projection method to solve linear systems in tensor format, Numerical linear algebra with applications 20 (1) (2013) 27–43. doi:10.1002/nla.1818.

[19] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, D. V. Savostyanov, Computation of extreme eigenvalues in higher dimensions using block tensor train format, Computer Physics Communications 185 (4) (2014) 1207–1216. doi:10.1016/j.cpc.2013.12.017.

[20] H. Rauhut, R. Schneider, Z. Stojanac, Tensor completion in hierarchical tensor representations, ArXiv e-prints.

[21] K. Reich, Die Entwicklung des Tensorkalküls: Vom absoluten Differentialkalkül zur Relativitätstheorie, Science Networks, Historical Studies, Birkhäuser Basel, 1994. doi:10.1007/978-3-0348-8486-0.

[22] H.-J. Dirschmid, Tensoren und Felder, Springer, 1996. doi:10.1007/978-3-7091-6589-8.

[23] W. R. Hamilton, On some extensions of quaternions, Philosophical Magazine 7 (1854) 492–499.

[24] W. Voigt, Die fundamentalen physikalischen Eigenschaften der Krystalle in elementarer Darstellung, Veit u. Co, 1898.

[25] E. B. Wilson, J. W. Gibbs, Vector analysis: A text-book for the use of students of mathematics & physics, Founded upon the lectures of J. W. Gibbs, Charles Scribner's Sons, 1901.

[26] C. F. Gauß, Disquisitiones generales circa superficies curvas, Typis Dieterichianis, 1828.

[27] E. B. Christoffel, Über die Transformation der homogenen Differentialausdrücke zweiten Grades, Journal für die reine und angewandte Mathematik 70 (1869) 46–70.

[28] G. Ricci-Curbastro, Résumé de quelques travaux sur les systèmes variables de fonctions associés à une forme différentielle quadratique, Bulletin des Sciences Mathématiques 16 (2) (1892) 167–189.

[29] M. M. G. Ricci, T. Levi-Civitai, Méthodes de calcul différentiel absolu et leurs applications, Mathematische Annalen 54 (1–2) (1900) 125–201. doi:10.1007/BF01454201.

[30] J. C. Maxwell, A treatise on electricity and magnetism, Vol. 1, Oxford, Clarendon Press, 1873.

[31] J. C. Maxwell, A treatise on electricity and magnetism, Vol. 2, Oxford, Clarendon Press, 1873.

[32] A. Einstein, Die Grundlage der allgemeinen Relativitätstheorie, Annalen der Physik 354 (1916) 769–822. doi:10.1002/andp.19163540702.

[33] F. L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, Journal of Mathematics and Physics 6 (1927) 164–189. doi:10.1002/sapm192761164.

[34] J. D. Carroll, J. J. Chang, Analysis of individual differences in multidimensional scaling via an n-way generalization of 'Eckart-Young' decomposition, Psychometrika 35 (3) (1970) 283–319. doi:10.1007/BF02310791.

[35] R. A. Harshman, Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis, UCLA Working Papers in Phonetics 16 (1970) 1–84.

[36] V. de Silva, L.-H. Lim, Tensor rank and the ill-posedness of the best low-rank approximation problem, SIAM Journal on Matrix Analysis and Applications 30 (3) (2008) 1084–1127. doi:10.1137/06066518X.

[37] S. Holtz, T. Rohwedder, R. Schneider, The alternating linear scheme for tensor optimization in the tensor train format, SIAM Journal on Scientific Computing 34 (2) (2012) A683–A713. doi:10.1137/100818893.

[38] L. R. Tucker, Implications of factor analysis of three-way matrices for measurement of change, in: C. W. Harris (Ed.), Problems in measuring change, University of Wisconsin Press, 1963, pp. 122–137.

[39] L. R. Tucker, The extension of factor analysis to three-dimensional matrices, in: H. Gulliksen, N. Frederiksen (Eds.), Contributions to mathematical psychology, Holt, Rinehart and Winston, 1964, pp. 110–127.

[40] M. H. Beck, A. Jäckle, G. A. Worth, H. D. Meyer, The multiconfiguration time-dependent Hartree (MCTDH) method: A highly efficient algorithm for propagating wavepackets, Physics Reports 324 (2000) 1–105. doi:10.1016/S0370-1573(99)00047-2.

[41] O. Koch, C. Lubich, Dynamical tensor approximation, SIAM Journal on Matrix Analysis and Applications 31 (5) (2010) 2360–2375. doi:10.1137/09076578X.

[42] I. V. Oseledets, Tensor-train decomposition, SIAM Journal on Scientific Computing 33 (5) (2011) 2295–2317. doi:10.1137/090752286.

[43] A. Arnold, T. Jahnke, On the approximation of high-dimensional differential equations in the hierarchical Tucker format, BIT Numerical Mathematics 54 (2) (2013) 305–341. doi:10.1007/s10543-013-0444-2.

[44] C. Lubich, T. Rohwedder, R. Schneider, B. Vandereycken, Dynamical approximation by hierarchical Tucker and tensor-train tensors, SIAM Journal on Matrix Analysis and Applications 34 (2) (2013) 470–494. doi:10.1137/120885723.

[45] I. Affleck, T. Kennedy, E. H. Lieb, H. Tasaki, Rigorous results on valence-bond ground states in antiferromagnets, Physical Review Letters 59 (7) (1987) 799–802. doi:10.1103/PhysRevLett.59.799.

[46] L. Grasedyck, D. Kressner, C. Tobler, A literature survey of low-rank tensor approximation techniques, GAMM-Mitteilungen 36 (1) (2013) 53–78. doi:10.1002/gamm.201310004.

[47] W. J. Anderson, Continuous-time Markov chains: An applications-oriented approach, Springer Series in Statistics, Springer New York, 2012. doi:10.1007/978-1-4612-3038-0.

[48] N. G. van Kampen, Stochastic processes in physics and chemistry, 3rd Edition, North-Holland Personal Library, Elsevier B.V., 2007. doi:10.1016/B978-044452965-7/50008-8.

[49] D. T. Gillespie, A general method for numerically simulating the stochastic time evolution of coupled chemical reactions, Journal of Computational Physics 22 (4) (1976) 403–434. doi:10.1016/0021-9991(76)90041-3.

[50] A. P. J. Jansen, An introduction to Monte Carlo simulations of surface reactions (2003). arXiv:cond-mat/0303028v1.

[51] M. J. Hoffmann, S. Matera, K. Reuter, kmos: A lattice kinetic Monte Carlo framework, Computer Physics Communications 185 (7) (2014) 2138–2150. doi:10.1016/j.cpc.2014.04.003.

[52] H. G. Winful, S. S. Wang, Stability of phase locking in coupled semiconductor laser arrays, Applied Physics Letters 53 (1988) 1894–1896. doi:10.1063/1.100363.

[53] J. B. Griffiths, The theory of classical dynamics, Cambridge University Press, 1985.

[54] Y. Qu, X. Duan, Progress, challenge and perspective of heterogeneous photocatalysts, Chemical Society Reviews 42 (7) (2013) 2568–2580. doi:10.1039/C2CS35355E.

[55] H. Gandhi, G. Graham, R. McCabe, Automotive exhaust catalysis, Journal of Catalysis 216 (1–2) (2003) 433–442. doi:10.1016/S0021-9517(02)00067-2.

[56] P. J. Schmid, J. L. Sesterhenn, Dynamic mode decomposition of numerical and experimental data, 61st Annual Meeting of the APS Division of Fluid Dynamics 53 (15) (2008) 208.

[57] P. J. Schmid, Dynamic mode decomposition of numerical and experimental data, Journal of fluid mechanics 656 (2010) 5–28. doi:10.1017/s0022112010001217.

[58] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, J. N. Kutz, On dynamic mode decomposition: Theory and applications, Journal of Computational Dynamics 1 (2) (2014) 391–421. doi:10.3934/jcd.2014.1.391.

[59] T. von Kármán, Aerodynamics: Selected topics in the light of their historical development, Dover Publications, 2004.

[60] C. Schütte, M. Sarich, Metastability and Markov state models in molecular dynamics: Modeling, analysis, algorithmic approaches, Courant Lecture Notes, Courant Institute of Mathematical Sciences, 2013.

[61] G. Froyland, G. A. Gottwald, A. Hammerlindl, A computational method to extract macroscopic variables and their dynamics in multiscale systems, SIAM Journal on Applied Dynamical Systems 13 (4) (2014) 1816–1846. doi:10.1137/130943637.

[62] S. Klus, P. Koltai, C. Schütte, On the numerical approximation of the Perron–Frobenius and Koopman operator, Journal of Computational Dynamics 3 (1) (2016) 51–79. doi:10.3934/jcd.2016003.

[63] L. D. Lathauwer, B. D. Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM Journal on Matrix Analysis and Applications 21 (4) (2000) 1253–1278. doi:10.1137/S0895479896305696.

[64] G. H. Golub, C. F. V. Loan, Matrix computations, 4th Edition, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, 2013.

[65] S. Friedland, V. Mehrmann, R. Pajarola, S. K. Suter, On best rank one approximation of tensors, Numerical Linear Algebra with Applications 20 (2013) 942–955. doi:10.1002/nla.1878.

[66] W. Hackbusch, Tensor spaces and numerical tensor calculus, Vol. 42 of Springer Series in Computational Mathematics, Springer, 2012. doi:10.1007/978-3-642-28027-6.

[67] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, SIAM Review 51 (3) (2009) 455–500. doi:10.1137/07070111X.

[68] O. Koch, C. Lubich, Dynamical tensor approximation, SIAM Journal on Matrix Analysis and Applications 31 (5) (2010) 2360–2375. doi:10.1137/09076578X.

[69] L. Grasedyck, Hierarchical singular value decomposition of tensors, SIAM Journal on Matrix Analysis and Applications 31 (4) (2010) 2029–2054. doi:10.1137/090764189.

[70] A. Falcó, W. Hackbusch, On minimal subspaces in tensor representations, Foundations of Computational Mathematics 12 (6) (2012) 765–803. doi:10.1007/s10208-012-9136-6.

[71] V. Kazeev, O. Reichmann, C. Schwab, Low-rank tensor structure of linear diffusion operators in the TT and QTT formats, Linear Algebra and its Applications 438 (11) (2013) 4204–4221. doi:10.1016/j.laa.2013.01.009.

[72] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218. doi:10.1007/BF02288367.

[73] B. N. Khoromskij, O(d log n)-quantics approximation of n-d tensors in high-dimensional numerical modeling, Constructive Approximation 34 (2) (2011) 257–280. doi:10.1007/s00365-011-9131-1.

[74] S. V. Dolgov, D. V. Savostyanov, Alternating minimal energy methods for linear systems in higher dimensions, SIAM Journal on Scientific Computing 36 (5) (2014) A2248–A2271. doi:10.1137/140953289.

[75] Y. Saad, Numerical methods for large eigenvalue problems, Society for Industrial and Applied Mathematics, 2011. doi:10.1137/1.9781611970739.

[76] T. Bouwmans, N. S. Aybat, E. Zahzah, Handbook of robust low-rank and sparse matrix decomposition: Applications in image and video processing, CRC Press, 2016.

[77] T. Rohwedder, A. Uschmajew, On local convergence of alternating schemes for optimization of convex problems in the tensor train format, SIAM Journal of Numerical Analysis 51 (2) (2013) 1134–1162. doi:10.1137/110857520.

[78] J. C. Butcher, Numerical methods for ordinary differential equations, 3rd Edition, John Wiley & Sons, 2016.

[79] A. Iserles, A first course in the numerical analysis of differential equations, Cambridge University Press, 2009.

[80] D. T. Gillespie, A rigorous derivation of the chemical master equation, Physica A 188 (1-3) (1992) 404–425. doi:10.1016/0378-4371(92)90283-V.

[81] R. F. Bass, Stochastic processes:, Cambridge University Press, 2011. doi:10.1017/CBO9780511997044.

[82] J. R. Norris, Markov chains, Cambridge University Press, 1997. doi:10.1017/CBO9780511810633.

[83] Z. Lin, Distributed control and analysis of coupled cell systems, VDM Verlag, 2008.

[84] W. Lenz, Beiträge zum Verständnis der magnetischen Eigenschaften in festen Körpern, Physikalische Zeitschrift 21 (1920) 613–615.

[85] E. Ising, Beitrag zur Theorie des Ferromagnetismus, Zeitschrift für Physik 31 (1925) 253–258.

[86] S. Lievens, N. Stoilova, J. V. der Jeugt, Harmonic oscillators coupled by springs: Discrete solutions as a Wigner quantum system, Journal of mathematical physics 47 (11) (2006) 113504. doi:10.1063/1.2364183.

[87] M. B. Plenio, J. Hartley, J. Eisert, Dynamics and manipulation of entanglement in coupled harmonic systems with many degrees of freedom, New Journal of Physics 6. doi:10.1088/1367-2630/6/1/036.

[88] N. Lee, A. Cichocki, Regularized computation of approximate pseudoinverse of large matrices using low-rank tensor train decompositions, SIAM J. Matrix Analysis Applications 37 (2) (2016) 598–623. doi:10.1137/15M1028479.

[89] B. W. Brunton, L. A. Johnson, J. G. Ojemann, J. N. Kutz, Extracting spatial-temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition, Journal of Neuroscience Methods 258 (2016) 1–15. doi:10.1016/j.jneumeth.2015.10.010.

[90] G. Froyland, P. K. Pollett, R. M. Stuart, A closing scheme for finding almost-invariant sets in open dynamical systems, Journal of Computational Dynamics 1 (1) (2014) 135–162. doi:10.3934/jcd.2014.1.135.

[91] M. O. Williams, I. G. Kevrekidis, C. W. Rowley, A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition, Journal of Nonlinear Science 25 (6) (2015) 1307–1346. doi:10.1007/s00332-015-9258-5.

[92] M. Budišić, R. Mohr, I. Mezić, Applied Koopmanism, Chaos: An Interdisciplinary Journal of Nonlinear Science 22 (4) (2012) 047510. doi:10.1063/1.4772195.

[93] G. Oster, A. Perelson, Chemical reaction networks, IEEE Transactions on Circuits and Systems 21 (6) (1974) 709–721. doi:10.1109/TCS.1974.1083946.

[94] M. Padidam, Chemically regulated gene expression in plants, Current opinion in plant biology 6 (2) (2003) 169–177. doi:10.1016/s1369-5266(03)00005-0.

[95] B. B. Aldridgea, J. M. Burke, D. A. Lauffenburger, P. K. Sorger, Physicochemical modelling of cell signalling pathways, Nature Cell Biology 8 (11) (2006) 1195–1203. doi:10.1038/ncb1497.

[96] S. K. Upadhyay, Chemical kinetics and reaction dynamics, Springer Netherlands, 2006. doi:10.1007/978-1-4020-4547-9.

[97] R. Srivastava, L. You, J. Summers, J. Yin, Stochastic vs. deterministic modeling of intracellular viral kinetics, Journal of Theoretical Biology 218 (2002) 309–321. doi:10.1006/yjtbi.3078.

[98] D. T. Gillespie, The chemical Langevin equation, Journal of Chemical Physics 113 (1) (2000) 297–306.

[99] R. Chang, Physical chemistry for the biosciences, University Science Books, Mill Valley, 2005.

[100] D. W. Oxtoby, H. P. Gillis, A. Campion, Principles of modern chemistry, 7th Edition, Cengage Learning, Independence, 2012.

[101] D. T. Gillespie, Exact stochastic simulation of coupled chemical reactions, Journal of Physical Chemistry 81 (25) (1977) 2340–2361. doi:10.1021/j100540a008.

[102] B. Munsky, M. Khammash, The finite state projection algorithm for the solution of the chemical master equation, The Journal of Chemical Physics 124 (4) (2006) 044104. doi:10.1063/1.2145882.

[103] M. Hegland, C. Burden, L. Santoso, S. MacNamara, H. Booth, A solver for the stochastic master equation applied to gene regulatory networks, Journal of Computational and Applied Mathematics 205 (2) (2007) 708–724. doi:10.1016/j.cam.2006.02.053.

[104] A. Ammar, E. Cueto, F. Chinesta, Reduction of the chemical master equation for gene regulatory networks using proper generalized decompositions, International Journal for Numerical Methods in Biomedical Engineering 28 (9) (2012) 960–973. doi:10.1002/cnm.2476.

[105] D. R. Herschbach, Molecular dynamics of elementary chemical reactions, in: B. G. Malmström (Ed.), Chemistry, 1981-1990, World Scientific, 1992, pp. 265–314.

[106] K. W. Kolasinski, Physical chemistry: How chemistry works, John Wiley & Sons, 2016.

[107] J. Kotz, P. Treichel, J. Townsend, Chemistry and chemical reactivity, Volume 2, Cengage Learning, 2008.

[108] D. L. Reger, S. R. Goode, D. W. Ball, Chemistry: Principles and practice, Available Titles OWL Series, Cengage Learning, 2009.

[109] K. Reuter, M. Scheffler, First-principles kinetic Monte Carlo simulations for heterogeneous catalysis: Application to the CO oxidation at $RuO_2(110)$, Physical Review B 73 (4) (2006) 045433. doi:10.1103/PhysRevB.73.045433.

[110] S. Matera, H. Meskine, K. Reuter, Adlayer inhomogeneity without lateral interactions: Rationalizing correlation effects in CO oxidation at $RuO_2(110)$ with first-principles kinetic Monte Carlo, Journal of Chemical Physics 134 (064713). doi:10.1063/1.3553258.

[111] G. J. Herschlag, S. Mitran, G. Lin, A consistent hierarchy of generalized kinetic equation approximations to the master equation applied to surface catalysis, The Journal of Chemical Physics 142 (23) (2015) 234703. doi:10.1063/1.4922515.

[112] J. Hagen, Industrial catalysis: A practical approach, John Wiley & Sons, 2015.

[113] G. Rothenberg, Catalysis: Concepts and green applications, Wiley-VCH Verlag GmbH & Co. KGaA, 2008. doi:10.1002/9783527621866.

[114] K. Reuter, First-principles kinetic Monte Carlo simulations for heterogeneous catalysis: Concepts, status, and frontiers, in: O. Deutschmann (Ed.), Modeling and Simulation of Heterogeneous Catalytic Reactions: From the Molecular Process to the Technical System, Wiley-VCH Verlag GmbH & Co. KGaA, 2011, pp. 71–112. doi:10.1002/9783527639878.ch3.

[115] B. Temel, H. Meskine, K. Reuter, M. Scheffler, H. Metiu, Does phenomenological kinetics provide an adequate description of heterogeneous catalytic reactions?, Journal of Chemical Physics 126 (20) (2007) 204711. doi:10.1063/1.2741556.

[116] M. Rieger, J. Rogal, K. Reuter, Effect of surface nanostructure on temperature programmed reaction spectroscopy: First-principles kinetic Monte Carlo simulations of CO oxidation at $RuO_2(110)$, Physical Review Letters 100 (1) (2008) 016105. doi:10.1103/PhysRevLett.100.016105.

[117] M. Stamatakis, D. G. Vlachos, Unraveling the complexity of catalytic reactions via kinetic Monte Carlo simulation: Current status and frontiers, ACS Catalysis 2 (12) (2012) 2648–2663. doi:10.1021/cs3005709.

[118] C. Kittel, Introduction to solid state physics, 8th Edition, John Wiley & Sons, Inc, 2004.

[119] H. Over, M. Muhler, Catalytic CO oxidation over ruthenium-bridging the pressure gap, Progress in Surface Science 72 (1–4) (2003) 3–17. doi:10.1016/S0079-6816(03)00011-X.

[120] K. Reuter, D. Frenkel, M. Scheffler, The steady state of heterogeneous catalysis, studied by first-principles statistical mechanics, Physical Review Letters 93 (11) (2004) 116105. doi:10.1103/PhysRevLett.93.116105.

[121] D. S. Sholl, J. A. Steckel, Density functional theory, John Wiley & Sons, Inc., 2009. doi:10.1002/9780470447710.ch1.

[122] E. Vanden-Eijnden, F. A. Tal, Transition state theory: Variational formulation, dynamical corrections, and error estimates, The Journal of Chemical Physics 123 (18) (2005) 184103. doi:10.1063/1.2102898.

[123] H. Meskine, S. Matera, M. Scheffler, K. Reuter, H. Metiu, Examination of the concept of degree of rate control by first-principles kinetic monte carlo simulations, Surface Science 603 (10) (2009) 1724–1730. doi:10.1016/j.susc.2008.08.036.

[124] S. Watanabe, Information theoretical analysis of multivariate correlation, IBM Journal of Research and Development 4 (1) (1960) 66–82. doi:10.1147/rd.41.0066.

[125] S. Kullback, R. A. Leibler, On information and sufficiency, The Annals of Mathematical Statistics 22 (1) (1951) 79–86. doi:10.1214/aoms/1177729694.

[126] S. Guerrero, E. E. Wolf, Monte Carlo simulation of stiff systems of catalytic reactions by sampling normally distributed rate probabilities, AIChE Journal 55 (11) (2009) 3022–3025. doi:10.1002/aic.11941.

[127] A. P. Gaylard, A. J. Baxendale, J. P. Howell, The use of CFD to predict the aerodynamic characteristics of simple automotive shapes, SAE Technical Paper (1998).

[128] A. M. Biadgo, A. Simonovič, J. Svorcan, S. Stupar, Aerodynamic characteristics of high speed train under turbulent cross winds: A numerical investigation using unsteady-RANS method, FME Transactions 42 (1) (2014) 10–18. doi:doi:10.5937/fmet1401010B.

[129] A. Goto, M. Zangeneh, Hydrodynamic design of pump diffuser using inverse design method and CFD, Journal of Fluids Engineering 124 (2) (2002) 319–328. doi:10.1115/1.1467599.

[130] L. Huilin, D. Gidaspow, Hydrodynamics of binary fluidization in a riser: CFD simulation using two granular temperatures, Chemical Engineering Science 58 (16) (2003) 3777–3792. doi:10.1016/S0009-2509(03)00238-0.

[131] S. Arulanandam, K. Hollands, E. Brundrett, A CFD heat transfer analysis of the transpired solar collector under no-wind conditions, Solar Energy 67 (1–3) (1999) 93–100. doi:10.1016/S0038-092X(00)00042-6.

[132] J. C. Han, S. Dutta, S. Ekkad, Gas turbine heat transfer and cooling technology, 2nd Edition, CRC Press, 2012.

[133] J. H. Ferziger, M. Peric, Computational methods for fluid dynamics, Springer Berlin Heidelberg, 2012.

[134] R. Eymard, T. Gallouët, R. Herbin, Finite volume methods, Handbook of numerical analysis 7 (2000) 713–1018.

[135] C. B. Laney, Computational gasdynamics, Cambridge University Press, 1998.

[136] D. Fultz, R. R. Long, G. V. Owens, W. Bohan, R. Kaylor, J. Weil, Studies of thermal convection in a rotating cylinder with some implications for large-scale atmospheric motions, American Meteorological Society, 1959. doi:10.1007/978-1-940033-37-2.

[137] T. von Larcher, C. Egbers, Experiments on transitions of baroclinic waves in a differentially heated rotating annulus, Nonlinear Processes in Geophysics 12 (6) (2005) 1033–1041. doi:10.5194/npg-12-1033-2005.

[138] U. Harlander, J. Wenzel, K. Alexandrov, Y. Wang, C. Egbers, Simultaneous piv and thermography measurements of partially blocked flow in a differentially heated rotating annulus, Experiments in Fluids 52 (4) (2012) 1077–1087. doi:10.1007/s00348-011-1195-y.

[139] T. von Larcher, A. Fournier, R. Hollerbach, The influence of a sloping bottom endwall on the linear stability in the thermally driven baroclinic annulus with a free surface, Theoretical and Computational Fluid Dynamics 27 (3) (2013) 433–451. doi:10.1007/s00162-012-0289-3.

[140] T. von Larcher, A. Dörnbrack, Numerical simulations of baroclinic driven flows in a thermally driven rotating annulus using the immersed boundary method, Meteorologische Zeitschrift 23 (6) (2015) 599–610. doi:10.1127/metz/2014/0609.

[141] J. M. Prusa, P. K. Smolarkiewicz, A. A. Wyszogrodzki, EULAG, a computational model for multiscale flows, Computers & Fluids 37 (9) (2008) 1193–1207. doi:10.1016/j.compfluid.2007.12.001.

[142] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, D. S. Henningson, Spectral analysis of nonlinear flows, Journal of Fluid Mechanics 641 (2009) 115–127.

[143] D. Duke, J. Soria, D. Honnery, An error analysis of the dynamic mode decomposition, Experiments in Fluids 52 (2) (2012) 529–542. doi:10.1007/s00348-011-1235-7.

[144] V. Kalro, T. Tezduyar, Parallel 3D computation of unsteady flows around circular cylinders, Parallel Computing 23 (9) (1997) 1235–1248. doi:10.1016/S0167-8191(97)00050-1.

[145] W. von Funck, T. Weinkauf, H. Theisel, H. P. Seidel, Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments, IEEE Transactions on Visualization and Computer Graphics 14 (6) (2008) 1396–1403. doi:10.1109/TVCG.2008.163.

[146] H. Jasak, A. Jemcov, Z. Tukovic, OpenFOAM: A C++ library for complex physics simulations, in: International workshop on coupled methods in numerical dynamics, 2007, pp. 1–20.

[147] D. S. Lemons, A. Gythiel, Paul Langevin's 1908 paper "On the theory of Brownian motion" ["Sur la théorie du mouvement brownien," C. R. Acad. Sci. (Paris) 146, 530–533 (1908)], American Journal of Physics 65 (11) (1997) 1079–1081. doi:10.1119/1.18725.

[148] T. Schlick, Molecular modeling and simulation: An interdisciplinary guide, Vol. 21, Springer New York, 2010. doi:10.1007/978-1-4419-6351-2.

[149] P. E. Kloeden, E. Platen, Numerical solution of stochastic differential equations, Stochastic Modelling and Applied Probability, Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-662-12616-5.

A Appendix

A.1. Proofs

A.1.1. Inverse Function for Little-Endian Convention (Lemma 2.4.1)

For this proof, we use the fact that

$$\sum_{k=1}^{c}(x_k-1)\prod_{l=1}^{k-1}n_l<\prod_{l=1}^{c}n_l, \tag{$*$}$$

for $c\in\mathbb{N}$ with $1\leq c\leq d$. This can be easily shown using mathematical induction. Here, we show that $\varphi_i=x_i$ for $i=1,\ldots,d$, which is also done using mathematical induction.

Basis:

From the definition of $\phi_N(x_1,\ldots,x_d)=\overline{x_1,\ldots,x_d}$ given in (2.4.1), we obtain for $i=d$

$$\begin{aligned}\varphi_d&=\left\lfloor\frac{\overline{x_1,\ldots,x_d}-1}{\prod_{l=1}^{d-1}n_l}\right\rfloor+1\\&=\left\lfloor\frac{\sum_{k=1}^{d}(x_k-1)\prod_{l=1}^{k-1}n_l}{\prod_{l=1}^{d-1}n_l}\right\rfloor+1\\&=\left\lfloor\underbrace{\frac{\sum_{k=1}^{d-1}(x_k-1)\prod_{l=1}^{k-1}n_l}{\prod_{l=1}^{d-1}n_l}}_{=:\alpha}+x_d-1\right\rfloor+1.\end{aligned}$$

From $(*)$ we know that $\alpha<1$. This implies

$$\begin{aligned}\varphi_d&=\lfloor\alpha+x_d-1\rfloor+1\\&=x_d-1+1\\&=x_d.\end{aligned}$$

Inductive step:

Assume the statement is true for an i with $1 < i \leq d$, i.e. $\varphi_i = x_i, \ldots, \varphi_d = x_d$. Using $(*)$, we obtain

$$\begin{aligned}
\varphi_{i-1} &= \left\lfloor \frac{\overline{x_1, \ldots, x_d} - 1 - \sum_{k=i}^{d} (\varphi_k - 1) \prod_{l=1}^{k-1} n_l}{\prod_{l=1}^{i-2} n_l} \right\rfloor + 1 \\
&= \left\lfloor \frac{\sum_{k=1}^{d} (x_k - 1) \prod_{l=1}^{k-1} n_l - \sum_{k=i}^{d} (x_k - 1) \prod_{l=1}^{k-1} n_l}{\prod_{l=1}^{i-2} n_l} \right\rfloor + 1 \\
&= \left\lfloor \frac{\sum_{k=1}^{i-2} (x_k - 1) \prod_{l=1}^{k-1} n_l}{\prod_{l=1}^{i-2} n_l} + x_{i-1} - 1 \right\rfloor + 1 \\
&= x_{i-1}.
\end{aligned}$$

□

A.1.2. Equivalence of Master Equation Formulations (Theorem 5.2.2)

Following the definition of the tensor multiplication, see e.g. [66], we can write

$$\begin{aligned}\left(\frac{\partial}{\partial t}\mathbf{P}(t)\right)_{x_1,\dots,x_d} &= \left(\left(\sum_{\mu=1}^{\mathcal{M}}(\mathbf{G}_\mu - \mathbf{I})\cdot\operatorname{diag}(\mathbf{a}_\mu)\right)\cdot\mathbf{P}(t)\right)_{x_1,\dots,x_d} \\ &= \sum_{\mu=1}^{\mathcal{M}}\sum_{y_1=1}^{n_1}\cdots\sum_{y_d=1}^{n_d}\left((\mathbf{G}_\mu - \mathbf{I})\cdot\operatorname{diag}(\mathbf{a}_\mu)\right)_{x_1,y_1,\dots,x_d,y_d}\cdot(\mathbf{P}(t))_{y_1,\dots,y_d}.\end{aligned}$$

Furthermore, it holds that

$$\begin{aligned}\left((\mathbf{G}_\mu - \mathbf{I})\cdot\operatorname{diag}(\mathbf{a}_\mu)\right)_{x_1,y_1,\dots,x_d,y_d} = &\sum_{z_1=1}^{n_1}\cdots\sum_{z_d=1}^{n_d}(\mathbf{G}_\mu)_{x_1,z_1,\dots,x_d,z_d} \\ &\cdot(\operatorname{diag}(\mathbf{a}_\mu))_{z_1,y_1,\dots,z_d,y_d} \\ &-\sum_{z_1=1}^{n_1}\cdots\sum_{z_d=1}^{n_d}(\mathbf{I})_{x_1,z_1,\dots,x_d,z_d} \\ &\cdot(\operatorname{diag}(\mathbf{a}_\mu))_{z_1,y_1,\dots,z_d,y_d}.\end{aligned}$$

Considering Definition 5.2.1 of the shift operators, this results in

$$(\operatorname{diag}(\mathbf{a}_\mu))_{x_1-\xi_\mu(1),y_1,\dots,x_d-\xi_\mu(d),y_d} - (\operatorname{diag}(\mathbf{a}_\mu))_{x_1,y_1,\dots,x_d,y_d}.$$

Just as $a_\mu(X)$ and $P(X,t)$ are set to zero if $X \notin \mathcal{S}$, we set

$$(\operatorname{diag}(\mathbf{a}_\mu))_{x_1-\xi_\mu(1),y_1,\dots,x_d-\xi_\mu(d),y_d} = 0,$$

if $x_k - \xi_\mu(k) \notin \{1,\dots,n_i\}$ for a $k \in \{1,\dots,d\}$. Analogously, we do the same for $(\mathbf{P}(t))_{x_1-\xi_\mu(1),\dots,x_d-\xi_\mu(d)}$. Due to the construction of $\operatorname{diag}(\mathbf{a}_\mu)$, we finally obtain

$$\left(\frac{\partial}{\partial t}\mathbf{P}(t)\right)_{x_1,\dots,x_d} = \sum_{\mu=1}^{\mathcal{M}} a_\mu(X-\xi_\mu)P(X-\xi_\mu,t) - a_\mu(X)P(X,t) = \frac{\partial}{\partial t}P(X,t). \quad \square$$

A.1.3. Equivalence of SLIM Decomposition and Canonical Representation (Theorem 6.2.1)

Consider the first two TT cores of the SLIM decomposition, given by

$$\begin{bmatrix}\mathbf{S}_1 & \mathbf{L}_1 & \mathbf{I}_1 & \mathbf{M}_1\end{bmatrix} \otimes \begin{bmatrix}\mathbf{I}_2 & 0 & 0 & 0\\ \mathbf{M}_2 & 0 & 0 & 0\\ \mathbf{S}_2 & \mathbf{L}_2 & \mathbf{I}_2 & 0\\ 0 & 0 & 0 & \mathbf{J}_2\end{bmatrix} =$$

$$\begin{bmatrix}\mathbf{S}_1 \otimes \mathbf{I}_2 + \mathbf{I}_1 \otimes \mathbf{S}_2 + [\mathbf{L}_1] \otimes [\mathbf{M}_2] & \mathbf{I}_1 \otimes [\mathbf{L}_2] & \mathbf{I}_1 \otimes \mathbf{I}_2 & [\mathbf{M}_1] \otimes [\mathbf{J}_2]\end{bmatrix}.$$

Successively, we obtain

$$\begin{aligned}\mathbf{A} = &\begin{bmatrix}\mathbf{S}_1 \otimes \mathbf{I}_2 \otimes \cdots \otimes \mathbf{I}_{d-1} + \ldots\\ \cdots + \mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_{d-2} \otimes \mathbf{S}_{d-1}\\ +[\mathbf{L}_1] \otimes [\mathbf{M}_2] \otimes \mathbf{I}_3 \otimes \cdots \otimes \mathbf{I}_{d-1} + \ldots\\ \cdots + \mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_{d-3} \otimes [\mathbf{L}_{d-2}] \otimes [\mathbf{M}_{d-1}]\\ \mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_{d-2} \otimes [\mathbf{L}_{d-1}]\\ \mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_{d-1}\\ [\mathbf{M}_1] \otimes [\mathbf{J}_2] \otimes \cdots \otimes [\mathbf{J}_{d-1}]\end{bmatrix}^{\mathbb{T}} \otimes \begin{bmatrix}\mathbf{I}_d\\ \mathbf{M}_d\\ \mathbf{S}_d\\ \mathbf{L}_d\end{bmatrix}\\ = &\mathbf{S}_1 \otimes \mathbf{I}_2 \otimes \cdots \otimes \mathbf{I}_d + \cdots + \mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_{d-1} \otimes \mathbf{S}_d\\ &+ [\mathbf{L}_1] \otimes [\mathbf{M}_2] \otimes \mathbf{I}_3 \otimes \cdots \otimes \mathbf{I}_d + \cdots + \mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_{d-2} \otimes [\mathbf{L}_{d-1}] \otimes [\mathbf{M}_d]\\ &+ [\mathbf{M}_1] \otimes [\mathbf{J}_2] \otimes \cdots \otimes [\mathbf{J}_{d-1}] \otimes [\mathbf{L}_d],\end{aligned}$$

which is exactly the same expression as (6.2.2). □

A.1.4. Equivalence of SLIM Decomposition and Canonical Representation for Markovian Master Equations (Theorem 6.3.1)

Consider the different parts of (6.3.7). It follows from the definitions given in (6.3.9), (6.3.10), and (6.3.11), that

$$\sum_{\nu=1}^{\alpha_i} \mathbf{A}_{i,\nu} = \mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_{i-1} \otimes \mathbf{S}_i \otimes \mathbf{I}_{i+1} \otimes \cdots \otimes \mathbf{I}_d,$$
$$\sum_{\mu=1}^{\beta_i} \mathbf{A}_{i,i+1,\mu} = \mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_{i-1} \otimes [\mathbf{L}_i] \otimes [\mathbf{M}_{i+1}] \otimes \mathbf{I}_{i+2} \otimes \cdots \otimes \mathbf{I}_d,$$

and

$$\sum_{\mu=1}^{\beta_d} \mathbf{A}_{d,1,\mu} = [\mathbf{M}_1] \otimes [\mathbf{J}_2] \otimes \cdots \otimes [\mathbf{J}_{d-1}] \otimes [\mathbf{L}_d]\,,$$

respectively. As we have shown in the proof of Theorem 6.2.1, see Appendix A.1.3, the SLIM decomposition given in (6.2.3) equals

$$\begin{aligned}
&\sum_{i=1}^{d} \mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_{i-1} \otimes \mathbf{S}_i \otimes \mathbf{I}_{i+1} \otimes \cdots \otimes \mathbf{I}_d \\
&\quad + \sum_{i=1}^{d-1} \mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_{i-1} \otimes [\mathbf{L}_i] \otimes [\mathbf{M}_{i+1}] \otimes \mathbf{I}_{i+2} \otimes \cdots \otimes \mathbf{I}_d \\
&\quad + [\mathbf{M}_1] \otimes [\mathbf{J}_2] \otimes \cdots \otimes [\mathbf{J}_{d-1}] \otimes [\mathbf{L}_d]\,.
\end{aligned}$$

Thus, the MME operator (6.3.7) can be represented by the SLIM decomposition with components of the form (6.3.9), (6.3.10), and (6.3.11). □

A.1.5. Functional Correctness of Pseudoinverse Algorithm (Theorem 7.2.1)

Since the left- and right-orthonormalization as well as the application of the SVD in step 2 of Algorithm 6 do not change the tensor $\mathbf{T}$ itself, we can express the matricization of $\mathbf{T}$ with respect to the dimensions $(1,\dots,l)$ and $(l+1,\dots,d)$ as

$$\mathbf{T}\Big|_{n_1,\dots,n_l}^{n_{l+1},\dots,n_d} = \tilde{U}\,\Sigma\,\tilde{V}^T,$$

with $\tilde{U}$, Σ, and $\tilde{V}$ as given in Algorithm 6. Now, we only have to show that $\tilde{U}^T\cdot\tilde{U} = \tilde{V}^T\cdot\tilde{V} = I \in \mathbb{R}^{r_l\times r_l}$. We obtain

$$\begin{aligned}\tilde{U}^T\cdot\tilde{U} = &\left(\left(\sum_{k_0=1}^{r_0}\cdots\sum_{k_{l-1}=1}^{r_{l-1}}\mathbf{T}^{(1)}_{k_0,:,k_1}\otimes\ldots\otimes\mathbf{T}^{(l)}_{k_{l-1},:,:}\right)\Big|_{n_1,\dots,n_l}^{r_l}\right)^T\\ &\cdot\left(\sum_{k'_0=1}^{r_0}\cdots\sum_{k'_{l-1}=1}^{r_{l-1}}\mathbf{T}^{(1)}_{k'_0,:,k'_1}\otimes\ldots\otimes\mathbf{T}^{(l)}_{k'_{l-1},:,:}\right)\Big|_{n_1,\dots,n_l}^{r_l}.\end{aligned}$$

Considering an entry of $\tilde{U}^T\cdot\tilde{U}$ and using (3.2.2), we then get

$$\begin{aligned}\left(\tilde{U}^T\cdot\tilde{U}\right)_{x,y} &= \left(\left(\sum_{k_0=1}^{r_0}\cdots\sum_{k_{l-1}=1}^{r_{l-1}}\mathbf{T}^{(1)}_{k_0,:,k_1}\otimes\ldots\otimes\mathbf{T}^{(l)}_{k_{l-1},:,x}\right)\Big|_{n_1,\dots,n_l}\right)^T\\ &\quad\cdot\left(\sum_{k'_0=1}^{r_0}\cdots\sum_{k'_{l-1}=1}^{r_{l-1}}\mathbf{T}^{(1)}_{k'_0,:,k'_1}\otimes\ldots\otimes\mathbf{T}^{(l)}_{k'_{l-1},:,y}\right)\Big|_{n_1,\dots,n_l}\\ &= \sum_{k_0=1}^{r_0}\cdots\sum_{k_{l-1}=1}^{r_{l-1}}\sum_{k'_0=1}^{r_0}\cdots\sum_{k'_{l-1}=1}^{r_{l-1}}\prod_{i=1}^{l}\left(\mathbf{T}^{(i)}_{k_{i-1},:,k_i}\right)^T\cdot\mathbf{T}^{(i)}_{k'_{i-1},:,k'_i},\end{aligned}$$

with $k_l = x$ and $k'_l = y$. Since $\mathbf{T}^{(1)}$ is left-orthonormal and $r_0 = 1$, we obtain $\left(\mathbf{T}^{(1)}_{1,:,k_1}\right)^T\cdot\mathbf{T}^{(1)}_{1,:,k'_1} = \delta_{k_1,k'_1}$. This implies that $\left(\tilde{U}^T\cdot\tilde{U}\right)_{x,y}$ is only nonzero if $k_1 = k'_1$. Now, we include the next core. This yields

$$\begin{aligned}&\sum_{k_1=1}^{r_1}\sum_{k'_1=1}^{r_1}\delta_{k_1,k'_1}\cdot\left(\mathbf{T}^{(2)}_{k_1,:,k_2}\right)^T\cdot\mathbf{T}^{(2)}_{k'_1,:,k'_2}\\ &\quad=\sum_{k_1=1}^{r_1}\left(\mathbf{T}^{(2)}_{k_1,:,k_2}\right)^T\cdot\mathbf{T}^{(2)}_{k_1,:,k'_2} = \left(\mathbf{T}^{(2)}_{:,:,k_2}\Big|_{r_1,n_1}\right)^T\cdot\mathbf{T}^{(2)}_{:,:,k'_2}\Big|_{r_1,n_1} = \delta_{k_2,k'_2}\end{aligned}$$

since $\mathbf{T}^{(2)}$ is also left-orthonormal. Successively, it then follows that for $\left(\tilde{U}^T \cdot \tilde{U}\right)_{x,y}$ to be nonzero that $k_i = k'_i$ for $i = 2, \ldots, l-1$. Thus, we obtain

$$\left(\tilde{U}^T \cdot \tilde{U}\right)_{x,y} = \sum_{k_{l-1}=1}^{r_{l-1}} \left(\mathbf{T}^{(l)}_{k_{l-1},:,x}\right)^T \cdot \mathbf{T}^{(l)}_{k_{l-1},:,y} = \left(\mathbf{T}^{(l)}_{:,:,x}\Big|_{r_{l-1},n_l}\right)^T \cdot \mathbf{T}^{(l)}_{:,:,y}\Big|_{r_{l-1},n_l}.$$

Note that $\mathbf{T}^{(l)}$ is also left-orthonormal due to the construction (see steps 3 and 4 of Algorithm 6) and therefore

$$\tilde{U}^T \cdot \tilde{U} = I \in \mathbb{R}^{r_l \times r_l}.$$

Analogously, it can be shown that $\tilde{V}^T \cdot \tilde{V} = I$ using the right-orthonormality of $\mathbf{T}^{(l+1)}, \ldots, \mathbf{T}^{(d)}$. It follows that the pseudoinverse calculated by Algorithm 6 satisfies the equations given in Definition 7.1.1, e.g.

$$\begin{aligned}
\mathbf{T}\Big|^{n_{l+1},\ldots,n_d}_{n_1,\ldots,n_l} \cdot \left(\mathbf{T}\Big|^{n_{l+1},\ldots,n_d}_{n_1,\ldots,n_l}\right)^+ \cdot \mathbf{T}\Big|^{n_{l+1},\ldots,n_d}_{n_1,\ldots,n_l} &= \tilde{U}\,\Sigma\,\tilde{V}^T \cdot \tilde{V}\,\Sigma^{-1}\,\tilde{U}^T \cdot \tilde{U}\,\Sigma\,\tilde{V}^T \\
&= \tilde{U}\,\Sigma\,\tilde{V}^T \\
&= \mathbf{T}\Big|^{n_{l+1},\ldots,n_d}_{n_1,\ldots,n_l}.
\end{aligned}$$

□

A.2. Algorithms

A.2.1. Orthonormalization of Tensor Trains

Algorithm 9 Left-orthonormalization of tensor trains

Input: Tensor train $\mathbf{T} \in \mathbb{R}^N$ with cores $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(d)}$ and ranks $r_0, \ldots, r_d$.
Output: Left-orthonormalized $\mathbf{T}$ with TT ranks $s_0, \ldots, s_d$, $s_i \leq r_i$.

1: Set $s_0 = s_d = 1$.
2: **for** $i = 1, ..., d-1$ **do**
3: $M = \mathcal{L}\left(\mathbf{T}^{(i)}\right)$.
4: Compute SVD of M, i.e. $M = U\,\Sigma\,V^T$ with $\Sigma \in \mathbb{R}^{s_i \times s_i}$, $s_i \leq r_i$.
5: Update $\mathbf{T}^{(i)} \in \mathbb{R}^{s_{i-1} \times n_i \times s_i}$ such that $\mathcal{L}\left(\mathbf{T}^{(i)}\right) = U$.
6: Define $W = \Sigma\,V^T \cdot \mathcal{R}\left(\mathbf{T}^{(i+1)}\right)$.
7: Update $\mathbf{T}^{(i+1)} \in \mathbb{R}^{s_i \times n_{i+1} \times r_{i+1}}$ such that $\mathcal{R}\left(\mathbf{T}^{(i+1)}\right) = W$.
8: **end for**

Algorithm 10 Right-orthonormalization of tensor trains

Input: Tensor train $\mathbf{T} \in \mathbb{R}^N$ with cores $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(d)}$ and ranks $r_0, \ldots, r_d$.
Output: Right-orthonormalized $\mathbf{T}$ with TT ranks $s_0, \ldots, s_d$, $s_i \leq r_i$.

1: Set $s_0 = s_d = 1$.
2: **for** $i = d, ..., 2$ **do**
3: $M = \mathcal{R}\left(\mathbf{T}^{(i)}\right)$.
4: Compute SVD of M, i.e. $M = U\,\Sigma\,V^T$ with $\Sigma \in \mathbb{R}^{s_{i-1} \times s_{i-1}}$, $s_{i-1} \leq r_{i-1}$.
5: Update $\mathbf{T}^{(i)} \in \mathbb{R}^{s_{i-1} \times n_i \times s_i}$ such that $\mathcal{R}\left(\mathbf{T}^{(i)}\right) = V^T$.
6: Define $W = \mathcal{L}\left(\mathbf{T}^{(i-1)}\right) \cdot U\,\Sigma$.
7: Update $\mathbf{T}^{(i-1)} \in \mathbb{R}^{r_{i-2} \times n_{i-1} \times s_i}$ such that $\mathcal{L}\left(\mathbf{T}^{(i+1)}\right) = W$.
8: **end for**

A.2.2. ALS for Systems of Linear Equations

Algorithm 11 ALS for Systems of Linear Equations

Input: Symmetric positive definite TT operator $\mathbf{A} \in \mathbb{R}^{N \times N}$, a right-hand side $\mathbf{U} \in \mathbb{R}^N$ and a right-orthonormal initial guess $\mathbf{T} \in \mathbb{R}^N$.
Output: Updated approximation $\mathbf{T}$ of the solution of $\mathbf{A} \cdot \mathbf{T} = \mathbf{U}$.

Set $\mathbf{L}_0^{\mathbf{A}} = \mathbf{L}_0^{\mathbf{U}} = \mathbf{R}_{d+1}^{\mathbf{A}} = \mathbf{R}_{d+1}^{\mathbf{U}} = 1$.
for $i = d, \ldots, 2$ **do**
    Compute $\mathbf{R}_i^{\mathbf{A}}$ and $\mathbf{R}_i^{\mathbf{U}}$ by using formulae (4.2.17) and (4.2.19).
end for
for $i = 1, \ldots, d-1$ **do**
    if $i > 1$ **then**
        Compute $\mathbf{L}_{i-1}^{\mathbf{A}}$ and $\mathbf{L}_{i-1}^{\mathbf{U}}$ by using formulae (4.2.16) and (4.2.18).
    end if
    Apply iterative solver to find the solution v of $A_i\, v = u_i$.
    Reshape v into a tensor $\mathbf{V} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$.
    Apply QR factorization, i.e. $\mathcal{L}(\mathbf{V}) = Q \cdot R$, cf. (4.2.12).
    Set $\mathbf{T}^{(i)}$ to a reshaped version of Q such that $\mathcal{L}\left(\mathbf{T}^{(i)}\right) = Q$.
end for
Compute $\mathbf{L}_{d-1}^{\mathbf{A}}$ and $\mathbf{L}_{d-1}^{\mathbf{U}}$.
for $i = d, \ldots, 1$ **do**
    if $i < d$ **then**
        Compute $\mathbf{R}_{i+1}^{\mathbf{A}}$ and $\mathbf{R}_{i+1}^{\mathbf{U}}$.
    end if
    Apply iterative solver to find the solution v of $A_i\, v = u_i$.
    Reshape v into a tensor $\mathbf{V} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$.
    if $i > 1$ **then**
        Apply QR factorization, i.e. $\mathcal{R}(\mathbf{V}) = R^T \cdot Q^T$, cf. (4.2.13).
        Set $\mathbf{T}^{(i)}$ to a reshaped version of Q^T such that $\mathcal{R}\left(\mathbf{T}^{(i)}\right) = Q^T$.
    else
        Set $\mathbf{T}^{(i)}$ to $\mathbf{V}$.
    end if
end for

A.2.3. MALS for Systems of Linear Equations

Algorithm 12 MALS for Systems of Linear Equations

Input: Symmetric positive definite TT operator $\mathbf{A} \in \mathbb{R}^{N \times N}$, a right-hand side $\mathbf{U} \in \mathbb{R}^N$ and a right-orthonormal initial guess $\mathbf{T} \in \mathbb{R}^N$.
Output: Updated approximation $\mathbf{T}$ of the solution of $\mathbf{A} \cdot \mathbf{T} = \mathbf{U}$.

Set $\mathbf{L}_0^{\mathbf{A}} = \mathbf{L}_0^{\mathbf{U}} = \mathbf{R}_{d+1}^{\mathbf{A}} = \mathbf{R}_{d+1}^{\mathbf{U}} = 1$.
for $i = d, \ldots, 3$ **do**
    Compute $\mathbf{R}_i^{\mathbf{A}}$ and $\mathbf{R}_i^{\mathbf{U}}$ by using formulae (4.2.17) and (4.2.19).
end for
for $i = 1, \ldots, d-1$ **do**
    if $i > 1$ **then**
        Compute $\mathbf{L}_{i-1}^{\mathbf{A}}$ and $\mathbf{L}_{i-1}^{\mathbf{U}}$ by using formulae (4.2.16) and (4.2.18).
    end if
    Apply iterative solver to find the solution v of $A_{i,i+1}\, v = u_{i,i+1}$.
    Reshape v into a tensor $\mathbf{V} \in \mathbb{R}^{r_{i-1} \times n_i \times n_{i+1} \times r_{i+1}}$.
    Apply (truncated) SVD, i.e. $\mathbf{V}\Big|_{r_{i-1},n_i}^{n_{i+1},r_{i+1}} = U\, \Sigma\, V^T$, cf. (4.2.14).
    Set $\mathbf{T}^{(i)}$ to a reshaped version of U such that $\mathcal{L}\left(\mathbf{T}^{(i)}\right) = U$.
end for
for $i = d-1, \ldots, 1$ **do**
    if $i < d-1$ **then**
        Compute $\mathbf{R}_{i+2}^{\mathbf{A}}$ and $\mathbf{R}_{i+2}^{\mathbf{U}}$.
    end if
    Apply iterative solver to find the solution v of $A_{i,i+1}\, v = u_{i,i+1}$.
    Reshape v into a tensor $\mathbf{V} \in \mathbb{R}^{r_{i-1} \times n_i \times n_{i+1} \times r_{i+1}}$.
    Apply (truncated) SVD, i.e. $\mathbf{V}\Big|_{r_{i-1},n_i}^{n_{i+1},r_{i+1}} = U\, \Sigma\, V^T$.
    Set $\mathbf{T}^{(i+1)}$ to a reshaped version of V^T such that $\mathcal{R}\left(\mathbf{T}^{(i+1)}\right) = V^T$.
end for
Replace $\mathbf{T}^{(1)}$ such that $\mathcal{L}\left(\mathbf{T}^{(1)}\right) = U \cdot \Sigma$.

A.2.4. ALS for Eigenvalue Problems

Algorithm 13 ALS for Eigenvalue Problems

Input: Symmetric TT operator $\mathbf{A} \in \mathbb{R}^{N \times N}$ and a right-orthonormal initial guess $\mathbf{T} \in \mathbb{R}^{N}$ in TT format.

Output: Approximation of the b smallest/largest eigenvalues $\Lambda_1, \ldots, \Lambda_b$ and corresponding eigentensors $\mathbf{T} \in \mathbb{R}^{N \times b}$ in BTT format.

Set $\mathbf{L}_0^{\mathbf{A}} = \mathbf{L}_0^{\mathbf{U}} = \mathbf{R}_{d+1}^{\mathbf{A}} = \mathbf{R}_{d+1}^{\mathbf{U}} = 1$.
for $i = d, \ldots, 2$ **do**
    Compute $\mathbf{R}_i^{\mathbf{A}}$ and $\mathbf{R}_i^{\mathbf{U}}$ by using formulae (4.2.17) and (4.2.19).
end for
for $i = 1, \ldots, d-1$ **do**
    if $i > 1$ **then**
        Compute $\mathbf{L}_{i-1}^{\mathbf{A}}$ and $\mathbf{L}_{i-1}^{\mathbf{U}}$ by using formulae (4.2.16) and (4.2.18).
    end if
    Apply iterative solver in order to find the b smallest/largest eigenpairs $(\lambda_1, v_1), \ldots, (\lambda_b, v_b)$ of A_i.
    Reshape $(v_1, \ldots, v_b) \in \mathbb{R}^{(r_{i-1} \cdot n_i \cdot r_i) \times b}$ into a tensor $\mathbf{V} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i \times b}$.
    Apply truncated QR factorization, i.e. compute $\mathbf{V}\Big|_{r_{i-1}, n_i}^{r_i, b} = Q \cdot R$ with $Q \in \mathbb{R}^{(r_{i-1} \cdot n_i) \times r_i}$.
    Set $\mathbf{T}^{(i)}$ to a reshaped version of Q such that $\mathcal{L}\left(\mathbf{T}^{(i)}\right) = Q$.
end for
Compute $\mathbf{L}_{d-1}^{\mathbf{A}}$ and $\mathbf{L}_{d-1}^{\mathbf{U}}$.
for $i = d, \ldots, 1$ **do**
    if $i < d$ **then**
        Compute $\mathbf{R}_{i+1}^{\mathbf{A}}$ and $\mathbf{R}_{i+1}^{\mathbf{U}}$.
    end if
    Apply iterative solver in order to find the b smallest/largest eigenpairs $(\lambda_1, v_1), \ldots, (\lambda_b, v_b)$ of A_i.
    Reshape $(v_1, \ldots, v_b) \in \mathbb{R}^{b \times (r_{i-1} \cdot n_i \cdot r_i)}$ into a tensor $\mathbf{V} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i \times b}$.
    if $i > 1$ **then**
        Apply truncated QR factorization, i.e. compute $\mathbf{V}\Big|_{n_i, r_i}^{b, r_{i-1}} = Q \cdot R$ with $Q \in \mathbb{R}^{(n_i \cdot r_i) \times r_{i-1}}$.
        Set $\mathbf{T}^{(i)}$ to a reshaped version of Q^T such that $\mathcal{R}\left(\mathbf{T}^{(i)}\right) = Q^T$.
    else
        Set $\mathbf{T}^{(i)}$ to $\mathbf{V}$ and $\Lambda_k = \lambda_k$ for $k = 1, \ldots, b$.
    end if
end for

A.2.5. MALS for Eigenvalue Problems

Algorithm 14 MALS for Eigenvalue Problems

Input: Symmetric TT operator $\mathbf{A} \in \mathbb{R}^{N \times N}$ and a right-orthonormal initial guess $\mathbf{T} \in \mathbb{R}^{N}$ in TT format.

Output: Approximation of the b smallest/largest eigenvalues $\Lambda_1, \ldots, \Lambda_b$ and corresponding eigentensors $\mathbf{T} \in \mathbb{R}^{N \times b}$ in BTT format.

Set $\mathbf{L}_0^{\mathbf{A}} = \mathbf{L}_0^{\mathbf{U}} = \mathbf{R}_{d+1}^{\mathbf{A}} = \mathbf{R}_{d+1}^{\mathbf{U}} = 1$.
for $i = d, \ldots, 3$ **do**
    Compute $\mathbf{R}_i^{\mathbf{A}}$ and $\mathbf{R}_i^{\mathbf{U}}$ by using formulae (4.2.17) and (4.2.19).
end for
for $i = 1, \ldots, d-1$ **do**
    if $i > 1$ **then**
        Compute $\mathbf{L}_{i-1}^{\mathbf{A}}$ and $\mathbf{L}_{i-1}^{\mathbf{U}}$ by using formulae (4.2.16) and (4.2.18).
    end if
    Apply iterative solver in order to find the b smallest/largest eigenpairs $(\lambda_1, v_1), \ldots, (\lambda_b, v_b)$ of $A_{i,i+1}$.
    Reshape the matrix $(v_1, \ldots, v_b) \in \mathbb{R}^{(r_{i-1} \cdot n_i \cdot n_{i+1} \cdot r_{i+1}) \times b}$ into a tensor $\mathbf{V} \in \mathbb{R}^{r_{i-1} \times n_i \times n_{i+1} \times r_{i+1} \times b}$.
    Apply (truncated) SVD, i.e. compute $\mathbf{V}\Big|_{r_{i-1}, n_i}^{n_{i+1}, r_{i+1}, b} = U \, \Sigma \, V^T$.
    Set $\mathbf{T}^{(i)}$ to a reshaped version of U such that $\mathcal{L}\left(\mathbf{T}^{(i)}\right) = U$.
end for
Compute $\mathbf{L}_{d-1}^{\mathbf{A}}$ and $\mathbf{L}_{d-1}^{\mathbf{U}}$.
for $i = d-1, \ldots, 1$ **do**
    if $i < d-1$ **then**
        Compute $\mathbf{R}_{i+1}^{\mathbf{A}}$ and $\mathbf{R}_{i+1}^{\mathbf{U}}$.
    end if
    Apply iterative solver in order to find the b smallest/largest eigenpairs $(\lambda_1, v_1), \ldots, (\lambda_b, v_b)$ of $A_{i,i+1}$.
    Reshape the matrix $(v_1, \ldots, v_b)^T \in \mathbb{R}^{b \times (r_{i-1} \cdot n_i \cdot n_{i+1} \cdot r_{i+1})}$ into a tensor $\mathbf{V} \in \mathbb{R}^{b \times r_{i-1} \times n_i \times n_{i+1} \times r_{i+1}}$.
    Apply (truncated) SVD, i.e. compute $\mathbf{V}\Big|_{r_{i-1}, n_i}^{n_{i+1}, r_{i+1}, b} = U \, \Sigma \, V^T$.
    Set $\mathbf{T}^{(i+1)}$ to a reshaped version of V^T such that $\mathcal{R}\left(\mathbf{T}^{(i+1)}\right) = V^T$.
end for
Replace $\mathbf{T}^{(1)}$ such that $\mathcal{L}\left(\mathbf{T}^{(1)}\right) = U \cdot \Sigma$ and set $\Lambda_k = \lambda_k$ for $k = 1, \ldots, b$.

A.2.6. Compression of Two-Dimensional TT Operators

Algorithm 15 Compression of two-dimensional TT operators

Input: TT operator $\mathbf{G} = \left[\mathbf{G}^{(1)}\right] \otimes \left[\mathbf{G}^{(2)}\right] \in \mathbb{R}^{m \times m \times n \times n}$ with TT cores $\mathbf{G}^{(1)} \in \mathbb{R}^{1 \times m \times m \times r}$ and $\mathbf{G}^{(2)} \in \mathbb{R}^{r \times n \times n \times 1}$.

Output: TT operator $\mathbf{H} = \left[\mathbf{H}^{(1)}\right] \otimes \left[\mathbf{H}^{(2)}\right]$ with TT rank $s \leq r$ and $\mathbf{H} = \mathbf{G}$.

1: Compute full tensor $\mathbf{G}$ and reshape it as a matrix $G \in \mathbb{R}^{(m \cdot m) \times (n \cdot n)}$.
2: Apply compact singular value decomposition, i.e. $G = U \Sigma V^T$ with $U \in \mathbb{R}^{(m \cdot m) \times s}$, $\Sigma \in \mathbb{R}^{s \times s}$, and $V \in \mathbb{R}^{(n \cdot n) \times s}$.
3: Define $\mathbf{H}^{(1)} \in \mathbb{R}^{1 \times m \times m \times s}$ by $\mathbf{H}^{(1)}_{1,x,y,k} = U_{\overline{x,y},k}$.
4: Define $\mathbf{H}^{(2)} \in \mathbb{R}^{s \times n \times n \times 1}$ by $\mathbf{H}^{(2)}_{k,x,y,1} = (\Sigma V^T)_{k,\overline{x,y}}$.

A.2.7. Construction of SLIM Decompositions for Markovian Master Equations

Algorithm 16 Construction of SLIM decompositions for MMEs

Input: *Single-cell reactions* (SCR)
For each cell Θ_i, $1 \leq i \leq d$, and every $\mathrm{R}_{i,\nu}$, $\nu = 1, \ldots, \alpha_i$, define the net change $p_{i,\nu} \in \mathbb{Z}$ (see (6.3.5)) and the vector $\mathrm{a}_{i,\nu} \in \mathbb{R}^{n_i}$ (see (6.3.2)) containing the values of the corresponding reaction propensity.
Two-cell reactions (TCR)
For each pair of cells Θ_i, Θ_{i+1}, $1 \leq i \leq d-1$ ($1 \leq i \leq d$ if cyclic), and every $\mathrm{R}_{i,i+1,\mu}$, $\mu = 1, \ldots, \beta_i$, define the net changes $p_{i,i+1,\mu}, q_{i,i+1,\mu} \in \mathbb{Z}$ (see (6.3.6)) and the matrix $\mathrm{a}_{i,i+1,\mu} \in \mathbb{R}^{n_i \times n_{i+1}}$ (see (6.3.3)) containing the values of the corresponding reaction propensity.

Output: SLIM decomposition of master equation operator $\mathbf{A}$ given in (6.2.3) and (6.2.5), respectively.

1: **for** $i = 1, \ldots, d$ **do**
2: Compute $\mathbf{S}_i = \sum_{\nu=1}^{\alpha_i} \left(G_i(-p_{i,\nu}) - I\right) \cdot \mathrm{diag}\left(\mathrm{a}_{i,\nu}\right)$ as defined in (6.3.9).
3: **end for**
4: **for** $i = 1, \ldots, d-1$ ($i = 1, \ldots, d$ if NNIS is cyclic) **do**
5: **for** $\mu = 1, \ldots, \beta_i$ **do**
6: Compute canonical representation of propensity $\mathrm{a}_{i,i+1,\mu}$, i.e. $\mathrm{a}_{i,i+1,\mu} = \sum_{k=1}^{r_{i,i+1,\mu}} \left(\mathrm{a}^{(1)}_{i,i+1,\mu}\right)_{k,:} \otimes \left(\mathrm{a}^{(2)}_{i,i+1,\mu}\right)_{k,:}$.
7: Compute $L_{i,\mu,k}$, $\tilde{L}_{i,\mu,k}$, $M_{i+1,\mu,k}$, and $\tilde{M}_{i+1,\mu,k}$ as defined in (6.3.8).
8: **end for**
9: Construct $\mathbf{L}_i$ and $\mathbf{M}_{i+1}$ as defined in (6.3.10) and (6.3.11).
10: Apply Algorithm 15 to $[\mathbf{L}_i] \otimes [\mathbf{M}_{i+1}]$ in order to compress the cores $\mathbf{L}_i$ and $\mathbf{M}_{i+1}$.
11: **end for**

A.3. Deutsche Zusammenfassung (German Summary)

In den letzten Jahren sind Tensorzerlegungen zu einem wichtigen Werkzeug sowohl für die mathematische Modellierung von hochdimensionalen Systemen als auch für die Approximation von hochdimensionalen Funktionen geworden. Tensorbasierte Methoden werden bereits in unterschiedlichsten Anwendungsgebieten erfolgreich eingesetzt. Wir betrachten Tensoren als eine Verallgemeinerung von Matrizen mit einer Vielzahl von Indizes. Die Zahl der Elemente eines solchen Tensors – und somit sein Speicherbedarf – wächst dabei exponentiell mit der Zahl der Dimensionen. Dieses Phänomen wird als *Fluch der Dimensionalität* bezeichnet. Das Interesse in Tensorzerlegungen wächst stetig, da unlängst entwickelte Tensorformate gezeigt haben, dass es möglich ist diesen Fluch zu umgehen und hochdimensional Systeme zu betrachten, welche vorher nicht mit konventionellen numerischen Methoden untersucht werden konnten. Typische Anwendungsbereiche umfassen das Lösen von linearen Gleichungssystemen, Eigenwertproblemen und gewöhnlichen wie auch partiellen Differentialgleichungen.

Die hier vorgestellten Methoden umfassen die tensorbasierte Darstellung von Markovschen Mastergleichungen, die Tensorzerlegung von linearen Operatoren bezüglich Nächste-Nachbarn-Interaktionen, die tensorbasierte Erweiterung der Dynamic Mode Decomposition und die Approximation des Perron-Frobenius-Operators. Dabei konzentrieren wir uns in dieser Arbeit auf das sogenannte Tensor-Train-Format. Unsere Experimente zeigen, dass wir mithilfe dieser Darstellung präzise Approximationen der Lösungen von linearen Gleichungssystemen und Eigenwertproblemen bestimmen können, um zum Beispiel stationäre Wahrscheinlichkeitsverteilungen zu berechnen. Im Vergleich zu klassischen Methoden ist es dabei möglich den Rechenaufwand und die damit verbundene Rechenzeit deutlich zu senken. Wir sind somit in der Lage, Einblicke in die Dynamiken und Strukturen von hochdimensionalen Systemen zu gewinnen. Unserer Auffassung nach, bilden die hier präsentierten Methoden einen weiteren Beitrag zu den Anwendungsmöglichkeiten von Tensorzerlegungen.

Diese Dissertation ist in drei Teile gegliedert. In Teil I erläutern wir das Grundkonzept von Tensorzerlegungen. Tensoren im Allgemeinen werden in Kapitel 2 und verschiedene Tensorformate werden in Kapitel 3 vorgestellt. In Kapitel 4 beleuchten wir die Lösungsmethoden für Optimierungsprobleme im TT-Format. Teil II widmet sich unseren oben beschriebenen Beiträgen zum Konzept von Tensorzerlegungen, siehe Kapitel 5 bis 7. Des Weiteren zeigen wir die ersten Schritte in Richtung der Approximation von Transferoperatoren und ihrer Eigenfunktionen in Kapitel 8. Die Leistungsfähigkeit von verschiedenen tensorbasierten Methoden wird in Teil III anhand mehrerer Beispiele aus unterschiedlichsten Anwendungsgebieten verdeutlicht. Wir betrachten chemische Reaktionsnetzwerke in Kapitel 9, heterogene katalytische Prozesse in Kapitel 10 und Beispiele aus dem Bereich der Fluid- und Molekülдynamik in Kapitel 11 bzw. 12. Den Abschluss dieser Arbeit bildet eine Zusammenfassung und ein Ausblick auf weitere Forschungsmöglichkeiten in Kapitel 13.

A.4. Eidesstattliche Erklärung (Declaration)

Ich versichere hiermit an Eides statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben. Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, den 24. April 2017

Patrick Gelß