Time Integration in Quantum Dynamics using Tree Tensor Networks

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
M. Sc. Dominik Sulz
aus Stuttgart

Tübingen
2024

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	13.12.2024
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Christian Lubich
2. Berichterstatter:	Prof. Dr. André Uschmajew

Abstract

This thesis studies the numerical solution of high-dimensional tensor differential equations. The prohibitive computational cost and memory requirements of numerically simulating such equations is often referred to as the *curse of dimensionality*. Such prohibitively large differential equations arise in many fields of applications such as plasma physics, machine learning, radiation transport or quantum physics. Dynamical low-rank approximation offers a promising ansatz to overcome the curse by representing the high-dimensional tensors in a low-rank format and solving a projected differential equation on a low-rank manifold. The low-rank manifold considered in this thesis is the manifold of tree tensor networks.

The time integration of tree tensor networks requires the update of each low-rank factor. Several numerical schemes to compute this time integration are proposed in this thesis. All of those methods fall into the class of Basis Update and Galerkin (BUG) integrators, where all basis matrices are evolved through a small matrix differential equation and all core tensors by a Galerkin step. We present a rigorous error analysis of all integration schemes that show robustness with respect to small singular values. This is important since small singular values can lead to numerical instabilities as they correspond to high curvatures in the corresponding low-rank manifold. Remarkable properties like rank-adaptivity, parallelism, norm and energy preservation and diminishing of energy in gradient systems are discussed.

Further, the representation of the right-hand side of a differential equation in tree tensor network format is discussed for a class of long-range interacting Hamiltonians. Efficient constructions of these tree tensor network operators are given and bounds on the maximal tree rank for an exact and approximated representation of the operator are proven. Numerical experiments for several problems from quantum physics verify theoretical results and investigate the applicability of dynamical low-rank approximation to many-body quantum systems in detail.

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der numerischen Lösung von hochdimensionalen Tensor-Differentialgleichungen. Die enormen Rechenkosten und Speicheranforderungen bei der numerischen Simulation solcher Gleichungen werden oft als "Fluch der Dimensionalität" bezeichnet. Solche hochdimensionalen Differentialgleichungen treten in vielen Anwendungsgebieten wie Plasmaphysik, maschinelles Lernen, Strahlentransport oder Quantenphysik auf. Ein vielversprechender Ansatz zur Überwindung dieses Fluchs ist die dynamische Niedrigrangapproximation, bei der hochdimensionale Tensoren in einem niedrigrangig Format dargestellt werden und dann eine projizierte Differentialgleichung auf einer Niedrigrang-Mannigfaltigkeit gelöst wird. Die in dieser Arbeit betrachtete Niedrigrang-Mannigfaltigkeit ist die Mannigfaltigkeit von Baum-Tensornetzwerken.
Die Zeitintegration von Baum-Tensornetzwerken erfordert die Zeitentwicklung jedes niederrang Faktors. In dieser Arbeit werden mehrere numerische Verfahren zur Berechnung dieser Zeitintegration vorgeschlagen. Alle diese Verfahren gehören zur Klasse der Basis-Update- und Galerkin-Integratoren (BUG), bei denen alle Basismatrizen durch eine kleine Matrixdifferentialgleichung und alle Kerntensoren durch einen Galerkin-Schritt entwickelt werden. Wir stellen eine rigorose Fehleranalyse aller Integrationsverfahren vor, die Robustheit gegenüber kleinen Singulärwerten aufweist. Dies ist wichtig, da kleine Singulärwerte zu numerischen Instabilitäten führen können, da sie hohen Krümmungen in der Niedrigrang-Mannigfaltigkeit entsprechen. Es werden bemerkenswerte Eigenschaften wie Rangadaptivität, Parallelität, Norm- und Energieerhaltung und Energieverringerung in Gradientensystemen diskutiert.
Des Weiteren wird die Darstellung der rechten Seite einer Differentialgleichung in Form eines Baum-Tensornetzwerks für eine Klasse von Hamiltonoperatoren mit langreichweitigen Wechselwirkungen diskutiert. Effiziente Konstruktionen dieser Baum-Tensornetzwerk Operatoren werden gegeben und Schranken für die maximalen Ränge für eine exakte und approximierte Darstellung des Operators werden bewiesen.
Numerische Experimente für mehrere Probleme aus der Quantenphysik verifizieren die theoretischen Ergebnisse und untersuchen die Anwendbarkeit der dynamischen Niedrigrangapproximation auf Vielteilchen-Quantensysteme im Detail.

Danksagungen

Zuallererst möchte ich mich bei meinem Betreuer Prof. Dr. Christian Lubich bedanken. Danke für das Ermöglichen dieser Arbeit und die exzellente Betreuung während meiner Promotion. Ich bin sehr dankbar für deine uneingeschränkte Unterstützung, das Beantworten unzähliger Fragen, unsere spannende Diskussionen und all das was ich in der Zeit von dir lernen durfte.

Inoltre, vorrei fare un ringraziamento speciale a Gianluca Ceruti. Non solo mi hai avvicinato al tema, ma mi hai anche sostenuto e incoraggiato durante tutta la promozione. Il tuo orecchio aperto e il tuo coinvolgimento hanno fatto la differenza nel successo di questo lavoro. Ti sono inoltre molto riconoscente per il soggiorno di ricerca ad Innsbruck.

Jonas Kusch und nochmals Gianluca Ceruti möchte ich für das Korrekturlesen dieser Arbeit und die wertvollen Anmerkungen dazu herzlich danken.

Außerdem geht mein Dank an die Numerikgruppe, insbesondere Gianluca Ceruti, Gabriele Eberspächer, Dominik Edelmann, Guilherme Fiusa, Ting Li, Christian Lubich, Jörg Nick, Yanyan Shi, Charlotte Verhoeven und Han Yang. Unsere Teerunden, gemeinsame Abendessen und die täglichen Abenteuer in der Mensa haben meinen Arbteitsalltag bereichert und mir immer Freude bereitet an die Uni zu kommen.

Ich möchte auch all meine Freunde erwähnen, die mich in dieser Zeit begleitet haben. Danke dass ihr ein so wichtiger Teil meines Lebens seid!

Mein letzter Dank geht an meine Familie: Mama, Papa und Aylin, danke dass ihr mich all die Zeit unterstützt habt. Dass ich diese Arbeit einreiche, verdanke ich zu einem großen Teil auch euch. Danke, dass ihr in allen Höhen und Tiefen immer für mich da wart.

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) through the Research Unit FOR 5413/1, Grant No. 465199066 and the Research Unit TRR 352, Project-ID 470903.

Contents

List of Figures

Chapter 1

Introduction

Long-range interacting quantum systems are currently a major research topic, as they govern the dynamics of future technologies [O'b07]. One highly relevant application is quantum computers, which are a disruptive technology that has the potential to lead to a significant change in industry, science and society. The advantage of quantum computers over classical computers is the possibility of creating superpositions and entanglement, offering the potential for significant computational speedups. Classical bits are restricted to binary values (0 and 1), while quantum bits (qubits) can exist in superpositions of these states, allowing for parallelism in computations. By this, problems like the unstructured search in data (Groover algorithm [Gro96]) or the prime factorization of large numbers (Shor's algorithm [Sho99]) can be solved much faster on a quantum computer. However, the quantum computers built so far are not large enough for practical computations, as their number of qubits is still too small. Consequently, simulating quantum circuits and their (long-range) interactions among qubits on classical computers remains a critical part of advancing quantum computing technology. The simulation and implementation of quantum circuits involves finding a solution of the high-dimensional Schrödinger equation

$$\dot{\psi}(t) = -iH\psi(t). \tag{1.1}$$

Here, H denotes a self-adjoint operator, called the Hamiltonian and $\psi(t) : \mathbb{R}^d \to \mathbb{C}$ denotes the d-dimensional wave function at time t. In many cases, the dimension d is large, which makes it costly or infeasible to solve the differential equation directly. The inherent high dimension of the numerical solution is not a problem only encountered in quantum physics. In fact, a vast number of problems in physics, engineering and computer science suffer from large memory requirements of numerical solutions.

In recent years, dynamical low-rank approximation (DLRA) has been successfully applied as an on-the-fly model order reduction technique to tackle the inherent curse of dimensionality. First introduced by Koch and Lubich in [KL07] for matrix and in [KL10] for tensor differential equations, the works laid the foundation for the development of stable time integration methods for matrix and tensor differential equations as the one in (1.1). The ansatz of DLRA applies to tensor differential equations of the form

$$\dot{A}(t) = F(t, A(t)), \quad A(t_0) = A_0, \tag{1.2}$$

where $A(t) \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ is a family of time-dependent tensors. Note that by setting $F = -iH$, the Schrödinger equation (1.1) emerges as one possible application for DLRA. The core idea of DLRA is to decompose the (possibly inaccessible) matrix or tensor $A(t)$ into a low-rank factorization and evolve all factors over time. The equations of motion for the factors are derived based on the Dirac-Frenkel time-dependent variational principle [Lub08], where the differential equation is projected onto a low-rank manifold. By assuming a low-rank approximability of the problem, significantly faster integration schemes can be derived. However, the equations of motion from [KL07, KL10] turn out to be stiff, since its Lipschitz constant becomes large in the presence of small singular values of $A(t)$. To overcome this issue, new robust integration schemes needed to be developed.

In 2014 Lubich and Oseledets presented the projector-splitting integrator for matrix differential equations [LO14]. It was the first numerical scheme based on DLRA which is robust in the presence of small singular values and therefore allows for numerical stable time integration. In 2021, the fixed-rank basis update and Galerkin (BUG) integrator for matrices was presented in [CL21] as another robust integrator. The fixed-rank BUG integrator enabled the derivation of several related integrators like the rank-adaptive BUG [CKL22], the parallel BUG [CKL24], the mid-point BUG [CEKL24] and the second-order parallel BUG [Kus24]. Other time integration schemes based on DLRA are based on a parallel in time ansatz [CGV23], a projected exponential method [CV23] and low-rank retractions [SCK24]. In [HNS23b, HNS23a] integration schemes for second-order matrix differential equations have been derived and in [NAB24] an implicit scheme for random partial differential equations is proposed.

The large variety of time integration schemes opened the possibility to influence many different research fields. Robust integrators for dynamical low-rank approximation have been successfully applied to (not exclusively) the following problems:

- In [PMF20, DEL19] the DLRA ansatz has been applied to radiation transport equations as well as in [KS23] to problems from radiation therapy, where the

radiation particles move through a background material. The latter can be used in the future for fast computations in treatment planning of lung cancer patients.

- Another application to a kinetic equation is the Vlasov-Poisson equation. It models the evolution of electrons in a collisionless plasma with an up to six-dimensional phase space. The above integrators have been successfully applied to these problems [EL18, EJ21, CE22, UZ24]. The stability of DLRA applications to hyperbolic problems is analyzed in [KEC23].

- DLRA has further influenced the training of neural networks. Instead of using a (stochastic) gradient descent method to update the parameters, one rewrites the update as a gradient system and applies the DLRA techniques. It was first done for neural networks in [SZK+22] and in [ZSC+23] for convolutional neural networks. Further applications on neural networks using DLRA can be found in e.g. [SZCT24, SHNT23].

- Further applications of DLRA have been the chemical master equation [EMP24b, EMP24a], stochastic and random differential equations [KNZ24, NT24] and uncertainty quantification [KCEF22].

In the wide range of applications for which DLRA holds results of interest, we focus on the quantum mechanic setting in this work. The first related applications can be found in [HLO+16, LOV15], where the authors applied a tensor version of the projector-splitting integrator to the Schrödinger equation (1.1). In [SLC+24] the authors applied a tensor version of the fixed-rank BUG integrator to an open quantum spin system. These works require the time integration of high-order tensors in a low-rank format. To address this inherent high dimensionality of the differential equation, different low-rank tensor representations exist and we refer to the survey articles [KB09, GKT13, BSU16, Bac23] for an in-depth discussion. The Tucker tensor format [Tuc66] is a widely used and practical tool for moderate ordered tensors, say tensors of order $d \leq 6$. Since in quantum mechanics much higher dimensions are of interest, tree tensor networks (TTNs) [CLW21] proved to be a promising ansatz to tackle this issue. TTNs are a data-sparse, hierarchical format to represent and approximate tensors and were first introduced in quantum chemistry, cf. [BJWM00, WT03] and the references therein. For the time integration of TTNs, the stable integrators (projector-splitting, BUG) have to be generalized to the TTN format. This has already been done for the projector-splitting integrator in [CLW21]. Note that there exists already software, which provides an implementation of the projector-splitting integrator for TTNs [MHM24a].

In the context of quantum mechanics, TTNs proved to be a promising ansatz, especially in the context of long-range interacting particles. General tree structures seem

to encode long-range correlations and interactions better than the tensor train/matrix products state format, the standard tool in computational quantum physics [SMC+23]. To bring all the advantages together, the main objectives of this thesis are to derive and analyze the fixed-rank BUG, the rank-adaptive BUG and the parallel BUG for tree tensor networks, represent long-range Hamiltonians in TTN format and apply all those methods to several problems from many-body quantum physics. Detailed contributions of this work are reported below.

1.1 Notation and conventions

Matrices

Throughout the thesis we write matrices in bold capitals. Since we are considering matrices with complex entries, we denote by $\mathbf{A}^\top$ the transpose of the matrix $\mathbf{A}$ and by $\mathbf{A}^*$ the transpose and complex conjugate of $\mathbf{A}$, i.e. $\mathbf{A}^* = \overline{\mathbf{A}}^\top$.

By $\bigotimes_{i=1}^d \mathbf{U}_i$ we denote the short notation of taking the Kronecker product of all the matrices $\mathbf{U}_i$, $i = 1, \dots, d$. Since the Kronecker product is not commutative, the order of the matrices in the product matters. We define it by

$$\bigotimes_{i=1}^{d} \mathbf{U}_i := \mathbf{U}_d \otimes \cdots \otimes \mathbf{U}_1.$$

When using norms of matrices we write $\|\cdot\|$ throughout the thesis if we use the Frobenius norm. The spectral norm will be denoted by $\|\cdot\|_2$. Thus, for $\mathbf{A} = (a_{k,l})_{k,l=1}^{n,m} \in \mathbb{C}^{n\times m}$ we have

$$\|\mathbf{A}\| = \sqrt{\sum_{k=1}^{n}\sum_{l=1}^{m} |a_{k,l}|^2}, \qquad \|\mathbf{A}\|_2 = \max_{\|x\|_2=1} \|\mathbf{A}x\|_2,$$

where $\|x\|_2$ denotes the Euclidean norm for a vector x.

Tensors

Throughout the thesis we write tensors in italic capitals. Let $A \in \mathbb{C}^{n_1\times\cdots\times n_d}$ be a tensor. Then we call the number of dimensions of A the order, i.e. A here is an order d tensor.

Analogously as for the matrix case, we denote by $\|A\|$ the Frobenius norm of a tensor $A \in \mathbb{C}^{n_1\times\cdots\times n_d}$, i.e.

$$\|A\| = \sqrt{\sum_{k_1=1}^{n_1} \cdots \sum_{k_d=1}^{n_d} |A(k_1, \dots, k_d)|^2}$$

In this thesis, we will present methods which change the rank of a matrix or tensor in a time step. To denote that a dimension of a tensor was augmented, we write $\widehat{r}$. I.e. for an order 3 tensor $A \in \mathbb{C}^{\widehat{r}_1 \times \widehat{r}_2 \times r_3}$, the notation means that the first two dimensions of A have been augmented.

1.2 Contributions of this thesis

We give an overview of the results of this thesis and their origins. Based on existing integration schemes for matrix and/or tensor differential equations, the thesis intends to extend these methods to the general class of tree tensor networks. We focus on the development and analysis of these time integration methods and further aim to provide all mathematical tools to apply these methods to problems arising in quantum physics. The analysis of these time-integration methods includes the proof of robust error bounds, which are independent of small singular values. The presented integrators allow for rank-adaptivity, such that optimal ranks regarding computational complexity and accuracy are chosen automatically. Further, the compact representation of operators of long-range interacting quantum systems in tree tensor network format is covered by the thesis. Finally, the time integration methods and strategies to construct operators are applied to several quantum systems. Especially long-range interacting quantum systems are challenging due to complex Hamiltonians and fast-growing correlations/entanglement in the system.

The main results were first published in the following four works of the same author, listed in chronological order of their first submission. Apart from the work "Numerical simulations of long-range open quantum many-body dynamics with tree tensor networks", the ordering of authorship is due to the mathematical tradition of alphabetical order. The other work uses the traditional ordering in physics. The authors are listed according to the degree of their contributions, with the last position reserved for the supervisor.

- Rank-adaptive time integration of tree tensor networks, SIAM Journal of Numerical Analysis 61 (2023), 194-222 [CLS23]. It is a joint work of Gianluca Ceruti, Christian Lubich and Dominik Sulz. The theoretical results of this work are equally distributed, while the numerical experiments are due to the author. Chapter 4 and Subsection 2.4.4 are mainly based on this work.

- Numerical simulations of long-range open quantum many-body dynamics with tree tensor networks, Physical Review A 109, 022420 [SLC+24]. It is a joint work of Dominik Sulz, Christian Lubich, Gianluca Ceruti, Igor Lesanovsky and Federico

Carollo. The theoretical results of this work are equally distributed, while the numerical experiments are due to Federico Carollo and the author. The numerical results from Section 6.3 are mainly based on this work.

- Low-Rank Tree Tensor Network Operators for Long-Range Pairwise Interactions, submitted [CKS24]. It is a joint work of Gianluca Ceruti, Daniel Kressner and Dominik Sulz. The theoretical results of this work are equally distributed, while the numerical experiments are due to the author. Chapter 3 is based on this work.

- Parallel Basis Update and Galerkin Integrator for Tree Tensor Networks, submitted [CKLS24]. It is a joint work of Gianluca Ceruti, Jonas Kusch, Christian Lubich and Dominik Sulz. The theoretical results of this work are equally distributed, while the numerical experiments are due to Jonas Kusch and the author. Chapter 5 is mainly based on this work.

The presentation of the results comes with a unified notation, which was already used in the [CLS23, CKLS24]. Moreover, much emphasis is put on the connection and interplay between the results from the different chapters. For example, the comparison of numerical results applying different time integration schemes to the same problem or the influence of approximated Hamiltonians on the dynamics, cf. chapter 6.

1.3 Outline

The results of the thesis are divided into the following five chapters.

Chapter 2: Tree Tensor Networks

This chapter recaps the formalism of tree tensor networks and discusses basic properties of those. To introduce the reader to this general class of tensors, we first introduce matrices (section 2.1) and the conceptually simpler Tucker tensors (section 2.2) and state some well-known results for those. In section 2.3 trees and tree tensor networks are defined. Section 2.4 explains basic operations with tree tensor networks like addition, orthogonalisation, contraction and truncation. For the latter, a rigorous error analysis is given. Since the algorithms in this thesis rely on projections onto manifolds of tensors, section 2.5 summarizes results about the differential geometry of several tensor manifolds. The last section 2.6 in this chapter introduces the Dirac-Frenkel time-dependent variational principle and motivates the development of robust time integration methods for tree tensor networks.

Chapter 3: Tree Tensor Network Operators

A fundamental ingredient when solving tensor differential equations by using tree tensor networks is to represent the operator of the right-hand side in the same format. The third chapter is therefore devoted to the construction of such operators for Hamiltonians encoding long-range interactions, which we call tree tensor network operators. Section 3.1 introduces the concept of tree tensor network operators and how one can apply those efficiently to a tree tensor network. Section 3.2 gives an explicit construction of a TTNO without further assumptions on the operator. Imposing some assumptions on a hierarchical low-rank structure enables the construction and approximation of more compact tree tensor network operators in section 3.3, including explicit bounds on the tree ranks. This is done by using hierarchical semi-separable matrices. The chapter closes with numerical examples of tree tensor network operator constructions of several operators coming from closed and open quantum spin systems in section 3.4.

Chapter 4: Rank-adaptive BUG integrator for Tree Tensor Networks

This chapter investigates the time integration of tree tensor networks by extending existing methods for matrices and Tucker tensors to the general class of tree tensor networks. Therefore, section 4.1 recaps a rank-adaptive integrator for Tucker tensors while in section 4.2 it is generalized first to extended Tucker tensors and in section 4.3 finally to tree tensor networks. In section 4.4 a fixed-rank variant of the tree tensor network integrator is discussed. The evolution of all nodes in a tree tensor network is done by solving small matrix/tensor differential equations. These differential equations require the construction of a reduced right-hand side operator and reduced initial data, which is presented in section 4.5. The remainder of the chapter is dedicated to proving interesting properties of the algorithm. Section 4.6 proves a preparatory lemma, which is needed to prove the other properties. Section 4.7 then states and proves an exactness property and a robust error bound, as well as a norm- and energy conservation property and the property of diminishing the energy for gradient systems.

Chapter 5: Parallel BUG integrator for Tucker Tensors and Tree Tensor Networks

The algorithm for time integration from chapter 4 allows for rank-adaptivity, but does not allow for a fully parallel structure. For large-scale computations, a fully parallel structure might become relevant, which is why in this chapter we derive a fully parallel method. The chapter starts with a recap of a parallel integrator for matrices in

seciton 5.1. In section 5.2 a parallel integrator for Tucker tensors is presented. Further, a robust error bound for this fully parallel Tucker integrator is proven. In a similar way as for the rank-adaptive method from the previous chapter, the fully parallel integrator is then formulated for tree tensor networks in section 5.3, together with a generalized robust error bound.

Chapter 6: Numerical experiments

The thesis closes with applications of the proposed methods to various differential equations. First, we verify theoretical predictions of the exactness property in section 6.1. The remaining examples stem from quantum physics/chemistry applications. We start by applying the integrators for tree tensor networks to quantum spin systems. Besides their interest in physics, they are also an adequate numerical test case, as no space discretization is needed. Hence, we can study the time and projection errors without any influence of space discretization. Generally, we consider systems with long- and short-range interacting particles. In section 6.2 we study closed quantum spin systems, while in section 6.3 an open quantum spin system is considered. The last example in section 6.4 stems from quantum molecular dynamics and investigates the Schrödinger equation with a Henon-Heiles potential, which is of interest in quantum chemistry.

Chapter 2

Tree Tensor Networks

Consider an initial value problem for a tensor differential equation of the from

$$\dot{A}(t) = F(t, A(t)), \quad A(t_0) = A_0, \tag{2.1}$$

where $A(t) \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ for all t and F is a given right-hand side. For large systems we need to find a good approximation $Y(t) \approx A(t)$. We aim to find the approximation $Y(t)$ in an approximation manifold $\mathcal{M}$ of smaller dimension. In the present thesis, we consider the manifold of tree tensor networks. Tree tensor networks are a hierarchical data-sparse format which allow to represent solutions to high-dimensional problems. A direct time integration of such large systems is typically impossible due the computational cost and memory requirement.

In this section we summarize the ideas and the theoretical background needed to work with tree tensor networks. In the following we will always consider a tensor $A \in \mathbb{C}^{n_1 \times \cdots \times n_d}$.

2.1 Matrices

Consider a simple case of a tensor, a matrix $\mathbf{A} \in \mathbb{C}^{n_1 \times n_2}$, with n_1, n_2 possibly very large. In section 2.3 we will see that matrices are a special case of a tree tensor network. The singular value decomposition decomposes $\mathbf{A}$ into orthogonal matrices $\mathbf{U}, \mathbf{V}$ and a diagonal matrix $\mathbf{S}$ with non-negative, real and decreasing entries, such that $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^*$, see for example [GVL13]. The diagonal elements of $\mathbf{S}$ are called singular values of $\mathbf{A}$.

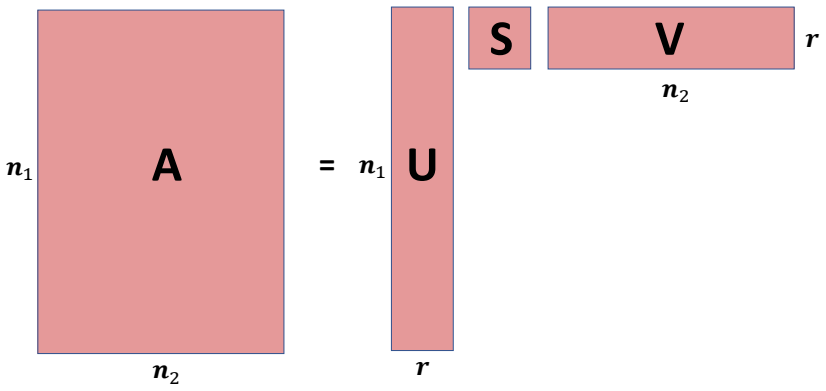

FIGURE 2.1: Singular value decomposition of a rank r matrix $\mathbf{A}$.

If $\mathbf{A}$ is of rank r it is known that only the first r singular values are non-zero [GVL13]. Hence $\mathbf{A}$ can be exactly decomposed by a reduced singular value decomposition

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^*, \ \mathbf{U} \in \mathbb{C}^{n_1 \times r}, \mathbf{V} \in \mathbb{C}^{n_2 \times r}, \ \mathbf{S} \in \mathbb{C}^{r \times r},$$

where $\mathbf{U}, \mathbf{V}$ are orthogonal and $\mathbf{S}$ contains only the first r singular vales on the diagonal. If $r \ll n_1, n_2$ the singular value decomposition is a data-sparse format to reduce the memory footprint, cf. figure 2.1 for a graphical illustration.

The singular value decomposition can be used to find low-rank approximations to possibly high/full-rank matrices. If $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^*$ is of rank k, we can easily find a rank $r < k$ approximation, which we denote by $\mathbf{A}_r$, by setting all singular vales $\sigma_{r+1} = \cdots = \sigma_k = 0$ and keeping only the first r columns of $\mathbf{U}$ and $\mathbf{V}$. $\mathbf{A}_r$ is then called the rank r approximation to $\mathbf{A}$. The error of this approximation is under control by the Eckart-Young theorem. It states that the truncated singular value decomposition actually gives the best approximation in the spectral norm, cf. [GVL13, Theorem 2.4.8].

Theorem 2.1: Eckart-Young

Let $\mathbf{A}$ be a matrix with $\operatorname{rank}(\mathbf{A}) = k$ and $\mathbf{A}_r$ be its rank r approximation obtained by the truncated singular value decomposition. Then

$$\min_{\operatorname{rank}(B)=r} \|\mathbf{A} - \mathbf{B}\|_2 = \|\mathbf{A} - \mathbf{A}_r\|_2 = \sigma_{r+1},$$

where $\|\cdot\|_2$ denotes the spectral norm for matrices.

Further, note that $\mathbf{A}_r$ is also the closest rank r matrix to $\mathbf{A}$ in the Forbenius norm.

2.2 Tucker Tensors

Suppose a vector v. To label the entries of v one index is sufficient, i.e. v_i denotes the ith entry if v. For a matrix $\mathbf{M}$, we need two indices M_{ij} to denote the entry in the ith row and jth column of $\mathbf{M}$. A tensor is now an object, where an arbitrary finite number of indices are allowed. Hence, a tensor is a multidimensional array of real or complex numbers, i.e. $A \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ or $A \in \mathbb{C}^{n_1 \times \cdots \times n_d}$. By this definition, vectors and matrices are included in the class of tensors. The number d is called the order of the tensor A. For a detailed introduction to tensors we refer to [Hac12].

2.2.1 Matricization and tensorization

The ith matricization of a tensor $A \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ is denoted by $\mathbf{Mat}_i(A) \in \mathbb{C}^{n_i \times n_{\neg i}}$, where $n_{\neg i} = \prod_{j \neq i}^{d} n_j$, cf. [CLW21, CLS23, KT14]. For the matricization the kth row aligns all entries of A that have k as the ith subscript. Usually, one orders reverse lexicographical. The inverse operation is called the tensorization and is denoted by Ten_i. It is clear that for a tensor A we have $A = Ten_i(\mathbf{A}_i)$ if and only if $\mathbf{A}_i = \mathbf{Mat}_i(A)$. Note that any ordering can be used for the matricization and tensorization, when it is done consistently.
For an illustration of the matricization suppose we have a tensor $A \in \mathbb{C}^{2 \times 3 \times 4}$, i.e. an order three tensor. Hence, we can matricize A in all these three modes. In figure 2.2 we illustrate all three matricizations of the tensor A graphically.

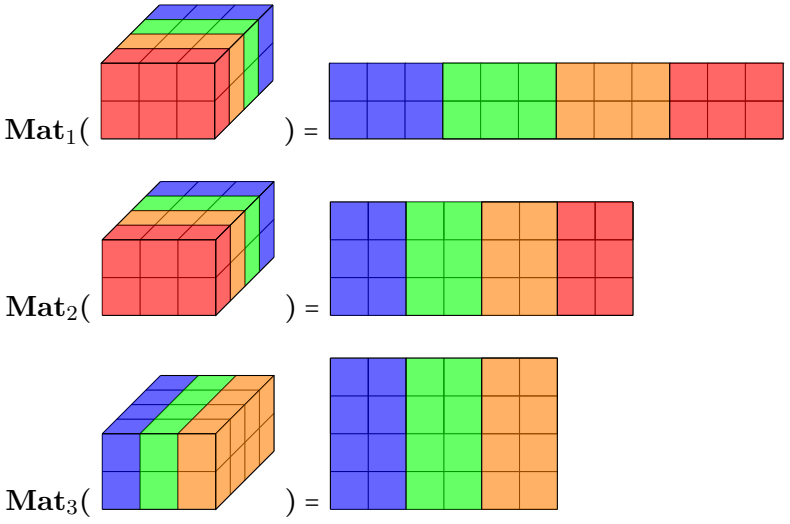

FIGURE 2.2: All matricizations of an order three tensor.

Matriczations will play a crucial role in the formulation and analysis of algorithms with tree tensor networks, as they allow to transform hard-to-handle objects like tensors into

matrices that are much easier to deal with. The next definition is in this spirit and defines the rank of a tensor by using the ranks of the matricized tensor.

Definition 2.1: Multilinear rank

Let $A \in \mathbb{C}^{n_1 \times \cdots \times n_d}$. The multilinear rank of the tensor A is defined as the d-tupel $(r_1, \ldots, r_d)$, where r_i is the rank of the matrix $\mathbf{Mat}_i(A)$, for $i = 1, ..., d$.

2.2.2 Tucker tensor format

For a tensor $C \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ and a matrix $\mathbf{U} \in \mathbb{C}^{n_i \times n_i}$ we define the tensor-matrix multiplication in the ith mode $C \times_i \mathbf{U}$ by (cf. [DDV00])

$$(C \times_i \mathbf{U})_{k_1,\ldots,k_i,\ldots,k_d} := \sum_{j=1}^{d} C_{k_1,\ldots,k_{i-1},j,k_{i+1},\ldots,k_d} (\mathbf{U}^\top)_{j,k_i}. \tag{2.2}$$

We know from [DDV00] that A has multilinear rank $(r_1, \ldots, r_d)$ if and only if A allows a so-called Tucker decomposition. A Tucker decomposition of a tensor $A \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ reads as

$$\mathbf{A} = C \bigtimes_{i=1}^{d} \mathbf{U}_i, \text{ where } a_{k_1,\ldots,k_d} = \sum_{l_1=1}^{r_1} \cdots \sum_{l_d=1}^{r_d} c_{l_1,\ldots,l_d} u^{(1)}_{k_1,l_1} \ldots u^{(d)}_{k_d,l_d}, \tag{2.3}$$

where $C \in \mathbb{C}^{r_1 \times \cdots \times r_d}$ is a tensor of full multilinear rank $(r_1, \ldots, r_d)$ and $\mathbf{U}_l = (u^{(l)}_{i,j})_{i,j} \in \mathbb{C}^{n_i \times r_i}$ are orthonormal matrices. It was originally developed by Ledyard R. Tucker in [Tuc66]. In literature, the Tucker decomposition is sometimes also referred to as the higher-order singular value decomposition (HOSVD). For tensors in Tucker format (or short Tucker tensor) we recall the unfolding formula from [KB09, Section 4]

$$\mathbf{Mat}_i \left(C \bigtimes_{j=1}^{d} \mathbf{U}_j \right) = \mathbf{U}_i \, \mathbf{Mat}_i(C) \bigotimes_{j \neq i} \mathbf{U}_j^\top. \tag{2.4}$$

Note that even for complex tensors formula (2.4) still holds. The unfolding formula (2.4) will be extremely useful in the analysis of tree tensor networks. Further, note that by the unfolding formula, the tensor-matrix multiplication from (2.2) can be rewritten by

$$C \times_i \mathbf{U} = Ten_i \left(\mathbf{U} \, \mathbf{Mat}_i(C) \right). \tag{2.5}$$

This statement is clear by setting the remaining matrices $\mathbf{U}_j$ to the identity and then applying the unfolding formula (2.4) to equation (2.5).

2.2.3 Quasi-best approximation for Tucker tensors

We again pose the question of whether it is possible to find the best approximation to a tensor, similar to the Eckart-Young theorem 2.1. Unlike in the matrix case, we will see that the best approximation property does not hold in the Tucker format. However, we obtain a, slightly weaker, quasi-best approximation, where the truncated Tucker tensor is the best-approximation up to a linear factor.
For that, following [DDV00] and [Gra10], we define the Tucker truncation by

Definition 2.2: Tucker truncation

Let $A \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ and

$$\mathbf{Mat}_i(A) = \widehat{\mathbf{U}}_i \widehat{\mathbf{S}}_i \widehat{\mathbf{V}}_i^*, \quad \text{for } i = 1, \dots, d,$$

a singular value decomposition with $\widehat{\mathbf{U}}_i \in \mathbb{C}^{n_i \times n_i}$ orthogonal and a diagonal matrix $\widehat{\mathbf{S}}_i = \operatorname{diag}(\sigma_{i,1}, \dots, \sigma_{i,n_i})$. Then the truncation of A to multilinear rank $(r_1, \dots, r_d)$ is defined by

$$\theta_{(r_1,\dots,r_d)}(A) := A \bigtimes_{i=1}^{d} \mathbf{U}_i \mathbf{U}_i^*,$$

where $\mathbf{U}_i$ is the matrix of the first r_i columns of $\widehat{\mathbf{U}}_i$.

Unlike the situation in the matrix case, where the svd-based truncation gave us the best approximation of a fixed rank (cf. theorem 2.1), in the Tucker case we do not obtain a best approximation statement. However, we obtain a quasi-optimality of the following form:

Theorem 2.2: Tucker truncation error

Let $A \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and denote the best-approximation in Tucker format of multi-dimensional rank $(r_1, \dots, r_d)$ by A^{best}. Then the error of the Tucker truncation is bounded by

$$\|A - \theta_{(r_1,\dots,r_d)}(A)\| \leq \sqrt{d} \|A - A^{\text{best}}\|.$$

Hence, up to a constant factor of the square root of d we obtain the best approximation with the Tucker truncation. This theorem and its proof can be found in [DDV00] while we used here the notation from [Gra10].

2.3 Tree Tensor Networks

Tucker tensors are a well-established tool for memory-efficient computations. For its core tensor $C \in \mathbb{C}^{r_1 \times \cdots \times r_d}$ assume for simplicity that $r_1 = \cdots = r_d =: r$ The memory footprint for C still scales like r^d, which becomes prohibitive if the order d is getting too large. Typically values such as $d > 6$ become problematic or inefficient. Nevertheless, there exist many problems where high-order tensors have to be considered, such as in quantum physics where the order d equals the number of particles/sites in the system. Tree tensor networks turned out to be a promising ansatz to overcome the so-called *curse of dimensionality* as they are an efficient hierarchical data sparse format to approximate tensors. Due to their recursive structure, they allow us to consider tensors of very high order while keeping the memory footprint at a reasonable size. Tree tensor networks have been considered first in the quantum chemistry literature in the framework of the multiconfiguration time-dependent Hartree (MCTDH) method and its multilayer variant, cf.[BJWM00, WT03] and the references therein.
In this thesis, we use the tree tensor network formalism introduced in [CLW21, CLS23]. The structure of a tree tensor network is encoded through an order tree, which is defined as follows [CLW21, Definition 2.1].

Definition 2.3: Ordered trees with unequal leaves

Let $\mathcal{L}$ be a finite set. The elements in $\mathcal{L}$ are referred to as leafs. We define $\mathcal{T}$ as the set of trees τ with the corresponding set of leaves $L(\tau) \subseteq \mathcal{L}$ recursively:

- Leafs are trees: $\mathcal{L} \subset \mathcal{T}$, and $L(l) := \{l\}$ for all $l \in \mathcal{L}$.
- Ordered m-tupels of trees are again trees, i.e. if for some $m \geq 2$,

$$\tau_1, \cdots, \tau_m \in \mathcal{T}, \text{ with } L(\tau_i) \cap L(\tau_j) = \varnothing \ \forall i \neq j,$$

then their ordered m-tupel is again in $\mathcal{T}$:

$$\tau := (\tau_1, \cdots, \tau_m) \in \mathcal{T} \quad \text{and} \quad L(\tau) := \dot{\bigcup}_{i=1}^{m} L(\tau_i).$$

In a graphical representation, this definition means that given trees $\tau_1, \ldots, \tau_m$, one obtains a new tree $\tau = (\tau_1, \ldots, \tau_m)$ by combining the subtrees $\tau_1, \ldots, \tau_m$ by taking a new node with m branches and connecting the ith branch with the subtree τ_i. This is graphically illustrated in figure 2.3.

On the set of trees we can define a partial ordering by setting for $\tau, \sigma \in \mathcal{T}$

- $\sigma \leq \tau$ if and only if σ is a subtree of τ,

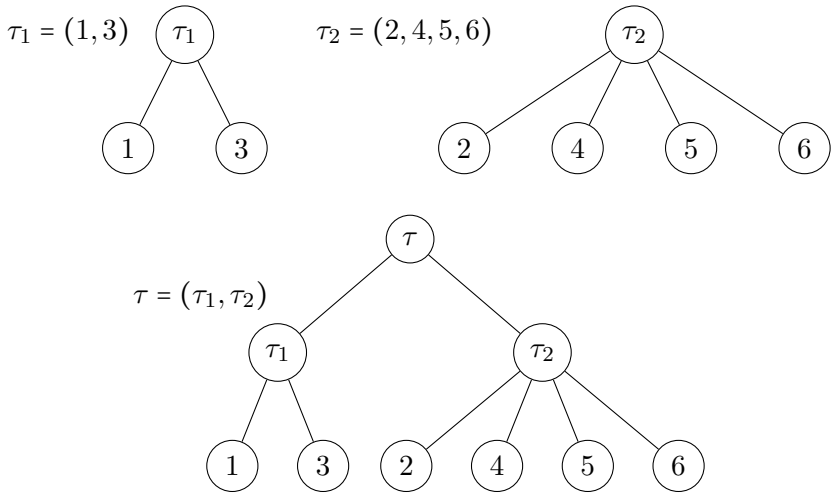

FIGURE 2.3: Graphical representation of a tree τ with two subtrees and the set of leaves $\mathcal{L} = \{1,2,3,4,5,6\}$.

- $\sigma < \tau$ if and only if σ is a subtree of τ and $\sigma \neq \tau$.

The height of a tree τ can be defined recursively:

- If $\tau = l$ is a leaf we set $h(l) = 0$.
- For a tree $\tau = (\tau_1, \ldots, \tau_m)$ we set $h(\tau) = \max\{h(\tau_1), \ldots, h(\tau_m)\} + 1$.

Therefore Tucker tensors are tree tensor networks of height 1, tensor trains/matrix product states are tree tensor networks of maximal height. Now that we have encoded the structure of a tree tensor network by a fixed tree $\bar{\tau} \in \mathcal{T}$, we can define what we understand by a tree tensor network. On the tree $\bar{\tau}$ we associate

- for each leaf $l \in \mathcal{L}$ a dimension n_l and a rank $r_l \leq n_l$ together with a so called basis matrix $\mathbf{U}_l \in \mathbb{C}^{n_l \times r_l}$ of rank r_l.
- for each subtree $\tau = (\tau_1, \ldots, \tau_m)$ a rank r_τ and a core tensor $C_\tau \in \mathbb{C}^{r_\tau \times r_{\tau_1} \times \cdots \times r_{\tau_m}}$ of multilinear rank $(r_\tau, r_{\tau_1}, \ldots, r_{\tau_m})$. Further we define $r_{\bar{\tau}} = 1$.

A necessary condition for C_τ, $\tau \leq \bar{\tau}$, to have multilinear rank $(r_\tau, r_{\tau_1}, \ldots, r_{\tau_m})$ is that with $r_\tau = r_{\tau_0}$ it holds

$$r_{\tau_i} \leq \prod_{j \neq i} r_{\tau_j}, \quad \text{for } i = 0, 1, \ldots, m. \tag{2.6}$$

This condition will be assumed throughout the whole thesis.

Note that the tree tensor network representation depends on the chosen tree structure $\bar{\tau}$. A fixed tensor allows for different representations by tree tensor networks if different trees are chosen. For example, an order three tensor can be equivalently represented by a Tucker tensor or by a hierarchical tree. With the considerations from above we define a tree tensor network as follows [CLW21, Definition 2.2].

Definition 2.4: Tree tensor network

For a given tree $\bar{\tau} \in \mathcal{T}$, basis matrices $\mathbf{U}_l$ and core tensors C_τ as described above, we recursively define a tensor $X_{\bar{\tau}}$ with a tree tensor network representation (or briefly a tree tensor network) as follows:

1. For each leaf $\tau = l \in \mathcal{L}$, we set

$$X_l := \mathbf{U}_l^\top \in \mathbb{C}^{r_l \times n_l} .$$

2. For each subtree $\tau = (\tau_1, \dots, \tau_m)$ (for some $m \geq 2$) of $\bar{\tau}$, we set $n_\tau = \prod_{i=1}^m n_{\tau_i}$ and $\mathbf{I}_\tau$ the identity matrix of dimension r_τ, and

$$X_\tau := C_\tau \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \mathbf{U}_{\tau_i} \in \mathbb{C}^{r_\tau \times n_{\tau_1} \times \cdots \times n_{\tau_m}},$$
$$\mathbf{U}_\tau := \mathbf{Mat}_0(X_\tau)^\top \in \mathbb{C}^{n_\tau \times r_\tau} .$$

   The subscript 0 in $\times_0$ and $\mathbf{Mat}_0(X_\tau)$ refers to the mode 0 of dimension r_τ in $\mathbb{C}^{r_\tau \times r_{\tau_1} \times \cdots \times r_{\tau_m}}$.

The tree tensor network $X_{\bar{\tau}}$ (more precisely, its representation in terms of the matrices $\mathbf{U}_\tau$) is called *orthonormal* if for each subtree $\tau < \bar{\tau}$, the matrix $\mathbf{U}_\tau$ has orthonormal columns.

The class of tree tensor networks includes matrices, Tucker tensors [Gra10, CL21, CKL22], hierarchical Tucker tenors [Gra10, KT14, Hac12] and tensor trains [Ose11, LOV15], which are also known as matrix product states (MPS) [CPGSV21, PGVWC07, VMC08] in the physical literature. Further, note that tree tensor networks are loop-free. In quantum physics tensor networks with loops, like projected entangled pair states (PEPS) [VMC08], are also used for quantum dynamical simulations. These tensor formats including loops are not covered by the class of tree tensor networks.

A graphical representation of a TTN has the same tree form as in figure 2.3. To distinguish between trees and tree tensor networks, we draw a TTN such that the core tensors are coloured in red and the physical dimension n_i is displayed at the ith leaf. In figure 2.4 several tree tensor network formats are illustrated

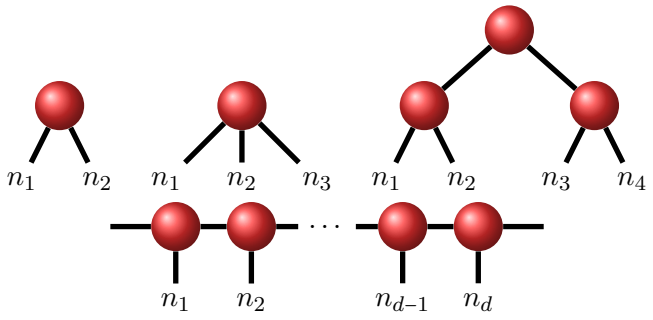

FIGURE 2.4: Graphical representation of several tree tensor networks. Top from left to right: Matrix, Tucker tensor, balanced binary TTN. Bottom: tensor train/matrix product state.

Finally, we want to relate the ranks of a tree tensor network with the ranks of the matricization of the corresponding full tensor. For this recall the following theorem from e.g. [Hac12, Theorem 11.12 and Lemma 11.15]:

Theorem 2.3

For a given tree $\bar{\tau}$, a tensor $X \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ admits a tree tensor network representation with

$$r_\tau = \operatorname{rank}\big(\mathbf{Mat}_{L(\tau)}(X)\big), \quad \forall \tau \leq \bar{\tau},$$

where the notation $\mathbf{Mat}_{L(\tau)}(X)$ denotes the matricization of X such that all indices of the modes in $L(\tau)$ are merged into row indices, while the remaining ones are merged into column indices.

2.4 Operations with Tree Tensor Networks

This section summarizes the main operations like addition, inner products, truncation and orthogonalization for tree tensor networks. These operations are needed in the course of this thesis to compute the dynamics of tree tensor networks.

2.4.1 Addition TTNs

The addition of TTN's follows the idea of the addition of hierarchical Tucker tensors (HT), cf. [KT14] and [Tob12, subsection 3.3.2]. We extend this idea to general, non-binary tree tensor networks.

To illustrate the idea, we start with two matrices in TTN form: $\mathbf{A} = \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^*$ and

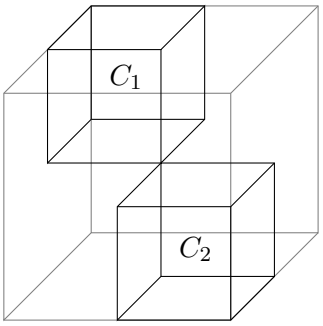

FIGURE 2.5: Block diagonal wise tensor embedding for two 3 dimensional tensors C_1, C_2.

$\mathbf{B} = \mathbf{U}_2\mathbf{S}_2\mathbf{V}_2^*$. Then the addition $\mathbf{D}$ can be implemented as an embedding by

$$\mathbf{D} := \mathbf{A} + \mathbf{B} = \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^* + \mathbf{U}_2\mathbf{S}_2\mathbf{V}_2^* = [\mathbf{U}_1, \mathbf{U}_2]\begin{bmatrix}\mathbf{S}_1 & \mathbf{0}\\ \mathbf{0} & \mathbf{S}_2\end{bmatrix}[\mathbf{V}_1, \mathbf{V}_2]^*.$$

Note that this strategy requires no arithmetic operations but an increase in storage cost. This embedding idea can be generalized to tree tensor networks. Let A and B be two TTNs defined on the same tree $\bar{\tau}$ and the same physical dimensions, i.e. $n_l^A = n_l^B,\ \forall l \in \mathcal{L}$. Note that the tree ranks $(r_\tau^A)_{\tau\leq\bar{\tau}}$, $(r_\tau^B)_{\tau\leq\bar{\tau}}$ can be different. The addition $D := A + B$ can be obtained as follows:

- Each leaf of the addition $D = A + B$ equals $[\mathbf{U}_l^1, \mathbf{U}_l^2]$, where $\mathbf{U}_l^1$ is the lth leaf of the TTN A and $\mathbf{U}_l^2$ is the lth leaf of the TTN B.
- The core tensor embedding is performed block diagonal wise. Suppose we have a core tensors $C_\tau^A \in \mathbb{C}^{r_1^A\times\cdots\times r_d^A}$ of A and the corresponding core tensor $C_\tau^B \in \mathbb{C}^{r_1^B\times\cdots\times r_d^B}$ of B. Then the core tensor of the addition D is a tensor $C_\tau \in \mathbb{C}^{r_1^A+r_1^B\times\cdots\times r_d^A+r_d^B}$, where

$$\begin{aligned}
&C_\tau(i_1,\ldots,i_d) = C_\tau^A(i_1,\ldots,i_d), && \text{if } 1 \leq i_j \leq r_j^A,\ \forall j = 1,\ldots,d.\\
&C_\tau(i_1,\ldots,i_d) = C_\tau^B(i_1 - r_1^A,\ldots,i_d - r_d^A), && \text{if } r_j^A < i_j \leq r_j^A + r_j^B,\ \forall j = 1,\ldots,d.\\
&C_\tau(i_1,\ldots,i_d) = 0, && \text{else.}
\end{aligned}$$

The embedding is graphically illustrated in figure 2.5. For the addition of TTN's all the core tensors of the addition are constructed by this embedding strategy. In the end, one obtains a TTN of bigger tree ranks $(r_\tau = r_\tau^A + r_\tau^B)_{\tau\leq\bar{\tau}}$. Performing an orthonormalization (see next subsection) to this increased TTN can reduce the tree ranks and therefore decrease the memory requirement.

2.4.2 Orthonormalization of TTNs

For applications it is favourable to work with orthonormal tree tensor networks. Each TTN has an orthonormal representation. In the following lemma, we proof that the orthonormality of a TTN localizes in the core tensors C_τ instead of the possibly huge matrices $\mathbf{U}_\tau$. The following lemma can be found in [CLW21, Lemma 2.4].

Lemma 2.1: Orthonormality of tree tensor networks

For a tree $\tau = (\tau_1, \dots, \tau_m) \in \mathcal{T}$, let the matrices $\mathbf{U}_{\tau_i}$ have orthonormal columns. Then the matrix $\mathbf{U}_\tau$ has orthonormal columns if and only if $\mathbf{Mat}_i(C_\tau)^\top$ has orthonormal columns.

Proof.
Recall that by definition 2.4, $\mathbf{U}_\tau = \mathbf{Mat}_0(X_\tau)^\top$ and the unfolding formula (2.4). Using this we obtain

$$\mathbf{U}_\tau^\top = \mathbf{Mat}_0(X_\tau) = \mathbf{I}_\tau \, \mathbf{Mat}_0(C_\tau) \bigotimes_{i=1}^{m} \mathbf{U}_{\tau_i}^\top.$$

From this we compute

$$\begin{aligned}\mathbf{U}_\tau^* \mathbf{U}_\tau &= \mathbf{I}_\tau \overline{\mathbf{Mat}_0(C_\tau)} \bigotimes_{i=1}^{m} \mathbf{U}_{\tau_i}^* \bigotimes_{i=1}^{m} \mathbf{U}_{\tau_i} \, \mathbf{Mat}_0(C_\tau)^\top \mathbf{I}_\tau \\ &= \overline{\mathbf{Mat}_0(C_\tau)} \left(\bigotimes_{i=1}^{m} \mathbf{U}_{\tau_i}^* \mathbf{U}_{\tau_i} \right) \mathbf{Mat}_0(C_\tau)^\top = (\mathbf{Mat}_0(C_\tau)^\top)^* \, \mathbf{Mat}_0(C_\tau)^\top,\end{aligned}$$

which proves the result. □

This lemma and the recursive definition of TTN's give us a strategy to orthogonalize a given tree tensor network. We perform multiple QR decompositions recursively from the leaves to the roots.

- If $\tau_i = l$ is a leaf with corresponding core tensor C_τ, we compute $\mathbf{U}_l = \mathbf{Q}_l \mathbf{R}_l$ and set $\mathbf{U}_l = \mathbf{Q}_l$ and $C_\tau = C_\tau \times_i \mathbf{R}_l$.
- At a core tensor C_{τ_i} we compute $\mathbf{Mat}_0(C_{\tau_i})^\top = \mathbf{Q}_{\tau_i} \mathbf{R}_{\tau_i}$ and set $C_{\tau_i} = Ten_0(\mathbf{Q}_{\tau_i}^\top)$ and $C_\tau = C_\tau \times_i \mathbf{R}_{\tau_i}$.

By this procedure the non-orthonormal components at each node are multiplied to the level above until we arrive at the root tensor. Then all matrices $\mathbf{U}_\tau$ for $\tau < \bar{\tau}$ have orthonormal columns.

2.4.3 Contraction products of TTNs

We define the contraction product of two tree tensor networks $X_\tau, Y_\tau \in \mathbb{C}^{r_\tau \times n_{\tau_1} \times \cdots \times n_{\tau_m}}$ by

$$\langle X_\tau, Y_\tau \rangle := \left(\mathbf{Mat}_0(X_\tau)^\top\right)^* \mathbf{Mat}_0(Y_\tau)^\top \in \mathbb{C}^{r_\tau \times r_\tau}. \tag{2.7}$$

In [CLW21, section 4.5] we see how this product can be computed in a recursive way, without ever computing the full tensors. Further, as $r_{\bar{\tau}} = 1$ the product $\langle X_{\bar{\tau}}, Y_{\bar{\tau}} \rangle \in \mathbb{C}$ and thus $\sqrt{\langle X_{\bar{\tau}}, X_{\bar{\tau}} \rangle}$ is the Euclidean norm of the orthonormal tree tensor network $X_{\bar{\tau}}$. The product can be computed efficiently by a recursion. Suppose we have two TTN's

$$X_\tau = C_\tau \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^{m} \mathbf{U}_{\tau_j}, \quad \mathbf{U}_{\tau_j} = \mathbf{Mat}_0(X_{\tau_j})^\top,$$
$$Y_\tau = D_\tau \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^{m} \mathbf{V}_{\tau_j}, \quad \mathbf{V}_{\tau_j} = \mathbf{Mat}_0(Y_{\tau_j})^\top.$$

With the unfolding formula (2.4) we obtain

$$\begin{aligned}
\langle X_\tau, Y_\tau \rangle &= \left(\mathbf{Mat}_0(X_\tau)^\top\right)^* \mathbf{Mat}_0(Y_\tau)^\top = \overline{\mathbf{Mat}_0(X_\tau)}\, \mathbf{Mat}_0(Y_\tau)^\top \\
&= \overline{\mathbf{Mat}_0(C_\tau)} \bigotimes_{j=1}^{m} \mathbf{U}_{\tau_j}^* \bigotimes_{j=1}^{m} \mathbf{V}_{\tau_j}\, \mathbf{Mat}_0(D_\tau)^\top \\
&= \overline{\mathbf{Mat}_0(C_\tau)} \bigotimes_{j=1}^{m} \left(\mathbf{U}_{\tau_j}^* \mathbf{V}_{\tau_j}\right) \mathbf{Mat}_0(D_\tau)^\top.
\end{aligned}$$

To proceed, the smaller products $\mathbf{U}_{\tau_j}^* \mathbf{V}_{\tau_j} = \langle X_{\tau_j}, Y_{\tau_j} \rangle$ can be computed via a recursion process. If we are at a leaf, i.e. $\mathbf{U}_{\tau_j} = \mathbf{U}_l$, we compute the matrix product $\mathbf{U}_l^* \mathbf{V}_l$, otherwise we use the prescribed recursion. Hence we only compute small matrix products and never an inaccessible tensor or basis matrix. The contraction procedure is illustrated in figure 2.6. The dashed lines correspond to the contraction of the connected basis matrices.

Using the addition and the contraction product, we can compute differences in norm between two tree tensor networks X and Y of the same tree structure. We compute $\|X - Y\|$ by multiplying Y with a constant factor of -1 and then add $D := X - Y$ by the addition from subsection 2.4.1. A multiplication of a scalar with a TTN is done by multiplying the root tensor of the TTN with the scalar. We then compute the norm via

$$\|X - Y\| = \|D\| = \sqrt{\langle D, D \rangle}, \tag{2.8}$$

where the inner product is computed as described in this subsection.

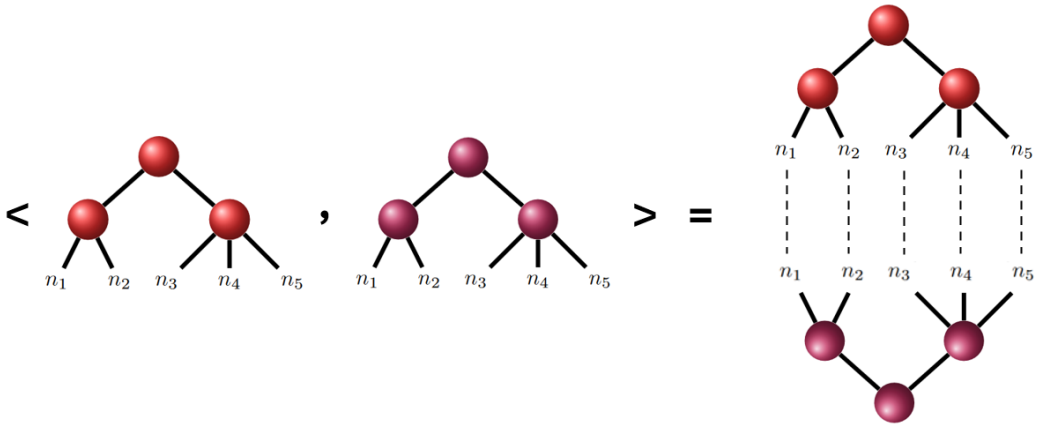

FIGURE 2.6: Graphical illustration of the contraction product.

2.4.4 Truncation of TTNs

The formulation and results of this subsection are published in [CLS23, appendix A] by the author in collaboration with Gianluca Ceruti and Christian Lubich.

As we will see in the course of this thesis (chapters 4 and 5), the ranks of a TTN are usually doubled after one time step. To keep the computation feasible while keeping a certain accuracy one needs to apply a rank-truncation with a given tolerance ϑ to the augmented tree tensor network. Nevertheless, the presented rank-truncation algorithm can be generally used for tree tensor networks.

The rank-truncation algorithm performs a recursive root-to-leaves truncation based on the singular value decomposition. In [Gra10] and [Hac12, Sec. 11.4.2] different rank truncation algorithms for binary tree tensor networks are studied. The here presented strategy and error analysis applies also to general (non-binary) tree tensor networks.

Algorithm 1: Rank truncation Θ_τ

Data: tree $\tau = (\tau_1, \dots, \tau_m)$, TTN in factorized form $\widehat{X}_\tau = \widehat{C}_\tau \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \widehat{\mathbf{U}}_{\tau_i}$
of tree rank $(\widehat{r}_\sigma)_{\sigma \le \tau}$, with $\widehat{\mathbf{U}}_{\tau_i} = \mathbf{Mat}_0(\widehat{X}_{\tau_i})^\top$ for a sub-TTN $\widehat{X}_{\tau_i}$,
tolerance parameter ϑ

Result: TTN in factorized form $X_\tau = C_\tau \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \mathbf{U}_{\tau_i}$
of tree rank $(r_\sigma)_{\sigma \le \tau}$ with adaptively chosen $r_\sigma \le \widehat{r}_\sigma$
with $\mathbf{U}_{\tau_i} = \mathbf{Mat}_0(X_{\tau_i})^\top$ for a rank-truncated sub-TTN X_{τ_i}

begin

  for $i = 1 : m$ in parallel **do**

    compute the reduced SVD $\mathbf{Mat}_i(\widehat{C}_\tau) = \widehat{\mathbf{P}}_{\tau_i} \mathbf{\Sigma}_{\tau_i} \widehat{\mathbf{Q}}^*_{\tau_i}$;

    set r_{τ_i} to be the smallest integer such that

$$\left(\sum_{k=r_{\tau_i}+1}^{\widehat{r}_{\tau_i}} \sigma_k^2 \right)^{1/2} \le \vartheta,$$

    where σ_k are the singular values in the diagonal matrix $\mathbf{\Sigma}_{\tau_i}$;

    set $\mathbf{P}_{\tau_i} \in \mathbb{C}^{\widehat{r}_{\tau_i} \times r_{\tau_i}}$ as the matrix of the first r_{τ_i} columns of $\widehat{\mathbf{P}}_{\tau_i}$;

    if $\tau_i = l$ is a leaf **then**

      set $\mathbf{U}_l = \widehat{\mathbf{U}}_l \mathbf{P}_l \in \mathbb{C}^{n_l \times r_l}$

    else

      set $\widetilde{C}_{\tau_i} = \widehat{C}_{\tau_i} \times_0 \mathbf{P}^\top_{\tau_i}$, where $\widehat{C}_{\tau_i}$ is the connection tensor of $\widehat{X}_{\tau_i}$;

      set $\widetilde{X}_{\tau_i}$ to be the TTN in which the connection tensor in $\widehat{X}_{\tau_i}$ is replaced with $\widetilde{C}_{\tau_i}$;

      compute $X_{\tau_i} = \Theta_{\tau_i}(\widetilde{X}_{\tau_i}, \vartheta)$ % `recursive truncation`

      set $\mathbf{U}_{\tau_i} = \mathbf{Mat}_0(X_{\tau_i})^\top$

    end

  end

  set $C_\tau = \widehat{C}_\tau \bigtimes_{i=1}^m \mathbf{P}^*_{\tau_i} \in \mathbb{C}^{r_\tau \times r_{\tau_1} \times \cdots \times r_{\tau_m}}$

end

Note that the resulting tree tensor network $X_{\bar{\tau}}$ is not orthonormal. If orthonormality is required one must apply the orthonormalization strategy from subsection 2.4.2 to $X_{\bar{\tau}}$. To prove an error bound for the presented rank truncation algorithm we assume an orthonormal tree tensor network $\widehat{X}^1_{\bar{\tau}}$, such that for each subtree $\tau = (\tau_1, \dots, \tau_m) \le \bar{\tau}$ the corresponding augmented tree tensor network has the form

$$\widehat{X}^1_\tau = \widehat{C}^1_\tau \times_0 \mathbf{I} \bigtimes_{i=1}^m \widehat{\mathbf{U}}^1_{\tau_i}, \quad \text{with} \quad \widehat{\mathbf{U}}_{\tau_i} = \mathbf{Mat}_0(\widehat{X}^1_{\tau_i})^\top$$

with $\left\|\widehat{\mathbf{U}}_\tau^1\right\|_2 = 1$ for all $\tau < \bar{\tau}$. In algorithm 1 we first perform a reduced singular value decomposition of the ith matricization of $\mathbf{Mat}_i(\widehat{C}_\tau^1)$. I.e. for $i = 1, \dots, m$ we compute

$$\mathbf{Mat}_i(\widehat{C}_\tau^1) = \widehat{\mathbf{P}}_{\tau_i} \widehat{\mathbf{S}}_{\tau_i} \widehat{\mathbf{Q}}_{\tau_i}^*, \tag{2.9}$$

where $\widehat{\mathbf{P}}_{\tau_i} \in \mathbb{C}^{\widehat{r}_{\tau_i} \times \widehat{r}_{\tau_i}}$ and $\widehat{\mathbf{Q}}_{\tau_i} \in \mathbb{C}^{\widehat{r}_{\tau_i} \times \neg\widehat{r}_{\tau_i}}$ are unitary matrices and $\widehat{\mathbf{S}}_{\tau_i} \in \mathbb{C}^{\widehat{r}_{\tau_i} \times \widehat{r}_{\tau_i}}$ is a diagonal matrix with decreasing and non-negative singular values. We rewrite the algorithm 1 in a mathematically equivalent formulation, which is easier to handle in the proof of the error bound. Therefore, we will define two tree tensor networks $\widehat{X}_\tau^{\mathrm{rot}}$ and $\widehat{X}_\tau^{\mathrm{cut}}$ recursively from the root to the leaves. To start the recursion, we set $\widehat{\mathbf{P}}_{\bar{\tau}} = 1$.

Rotation For a tree $\tau = (\tau_1, \dots, \tau_m)$ we define

$$\widehat{C}_\tau^{\mathrm{rot}} := \widehat{C}_\tau \times_0 \widehat{\mathbf{P}}_\tau^\top \bigtimes_{i=1}^m \widehat{\mathbf{P}}_{\tau_i}^* \quad \text{and} \quad \mathbf{U}_{\tau_i}^{\mathrm{rot}} := \mathbf{U}_{\tau_i} \widehat{\mathbf{P}}_{\tau_i},$$

for $i = 1, \dots, m$. Using the unfolding formula (2.4) and equation (2.9) we obtain

$$\mathbf{Mat}_i(\widehat{C}_\tau^{\mathrm{rot}}) = \widehat{\mathbf{P}}_{\tau_i}^* \, \mathbf{Mat}_i(\widehat{C}_\tau) \left(\bigotimes_{j \neq i} (\widehat{\mathbf{P}}_{\tau_j}^*)^\top \otimes \widehat{\mathbf{P}}_\tau \right) = \widehat{\mathbf{S}}_{\tau_i} \widehat{\mathbf{Q}}_{\tau_i}^* \left(\bigotimes_{j \neq i} (\widehat{\mathbf{P}}_{\tau_j}^*)^\top \otimes \widehat{\mathbf{P}}_\tau \right).$$

For the rotated basis matrix we get again by the unfolding formula (2.4)

$$\mathbf{U}_{\tau_i}^{\mathrm{rot}} = \mathbf{U}_{\tau_i} \widehat{\mathbf{P}}_{\tau_i} = \left(\widehat{\mathbf{P}}_{\tau_i}^\top \mathbf{Mat}_0(\widehat{X}_{\tau_i}) \right)^\top = \mathbf{Mat}_0 \left(\widehat{X}_{\tau_i} \times_0 \widehat{\mathbf{P}}_{\tau_i}^\top \right)^\top.$$

Using the derived equations form above we define

$$\begin{aligned} \widehat{X}_\tau^{\mathrm{rot}} &:= \widehat{X}_\tau \times_0 \mathbf{P}_\tau^\top = \widehat{C}_\tau \times_0 \mathbf{P}_\tau^\top \bigtimes_{i=1}^m \mathbf{U}_{\tau_i} \widehat{\mathbf{P}}_{\tau_i} \widehat{\mathbf{P}}_{\tau_i}^* \\ &= \left(\widehat{C}_\tau \times_0 \widehat{\mathbf{P}}_\tau^\top \bigtimes_{i=1}^m \widehat{\mathbf{P}}_{\tau_i}^* \right) \times_0 \mathbf{I}_{\widehat{r}_\tau} \bigtimes_{i=1}^m \mathbf{U}_{\tau_i} \widehat{\mathbf{P}}_{\tau_i} = \widehat{C}_\tau^{\mathrm{rot}} \times_0 \mathbf{I}_{\widehat{r}_\tau} \bigtimes_{i=1}^m \widehat{\mathbf{U}}_{\tau_i}^{\mathrm{rot}}. \end{aligned}$$

Thus, we have by the recursion that $\widehat{\mathbf{U}}_{\tau_i}^{\mathrm{rot}} = \mathbf{Mat}_0(\widehat{X}_{\tau_i}^{\mathrm{rot}})^\top$. We see that for $i = 1, \dots, m$

$$\mathbf{Mat}_i(\widehat{C}_\tau^{\mathrm{rot}}) = \widehat{\mathbf{S}}_{\tau_i} \widehat{\mathbf{V}}_{\tau_i}^* \tag{2.10}$$

for a matrix $\widehat{\mathbf{V}}_{\tau_i}$ with orthogonal columns. Further, we have $\widehat{X}_{\bar{\tau}}^{\mathrm{rot}} = \widehat{X}_{\bar{\tau}}$.

Cutting As in algorithm 1 we define the reduced rank r_{τ_i} as the smallest integer such that

$$\left(\sum_{k=r_{\tau_i}+1}^{\widehat{r}_{\tau_i}} \sigma_k^2 \right)^{1/2} \leq \vartheta, \tag{2.11}$$

where σ_k are the singular values in $\widehat{\mathbf{S}}_{\tau_i}$ and $\vartheta > 0$ is a given tolerance. With $\mathbf{S}_{\tau_i}$ we denote the truncated diagonal matrix with the largest r_{τ_i} singular values of $\widehat{\mathbf{S}}_{\tau_i}$. Then we define the cut core tensor $\widehat{C}_\tau^{\mathrm{cut}}$ of the same size as $\widehat{C}_\tau$ entry wise by

$$\widehat{C}_\tau^{\mathrm{cut}}(k_0, k_1, \ldots, k_m) := \begin{cases} \widehat{C}_\tau^{\mathrm{rot}}(k_0, k_1, \ldots, k_m) & \text{if } k_0 \leq r_\tau, k_i \leq r_{\tau_i} \text{ for } i = 1, \ldots, m, \\ 0 & \text{else.} \end{cases}$$

By this we see

$$\mathbf{Mat}_i(\widehat{C}_\tau^{\mathrm{cut}}) = \begin{pmatrix} \mathbf{S}_{\tau_i} & 0 \\ 0 & 0 \end{pmatrix} \widehat{\mathbf{V}}_{\tau_i}^*, \tag{2.12}$$

where $\mathbf{S}_{\tau_i}$ contains the first r_{τ_i} singular values of $\widehat{\mathbf{S}}_{\tau_i}$ and the remaining ones are set to zero. We then approximate the tree tensor network $\widehat{X}_\tau^{\mathrm{rot}}$ by the tree tensor network

$$\widehat{X}_\tau^{\mathrm{cut}} := \widehat{C}_\tau^{\mathrm{cut}} \times_o \mathbf{I}_\tau \bigtimes_{i=1}^{m} \widehat{\mathbf{U}}_\tau^{\mathrm{rot}}.$$

This expression can be further simplified by cutting all zero elements in the core tensor $\widehat{C}_\tau^{\mathrm{cut}}$. Therefore, we define the core tensor $C_\tau \in \mathbb{C}^{r_\tau \times r_{\tau_1} \times \cdots \times r_{\tau_m}}$ via

$$C_\tau(k_0, k_1, \ldots, k_m) := \widehat{C}_\tau^{\mathrm{cut}}(k_0, k_1, \ldots, k_m) \quad \text{for } k_0 \leq r_\tau, k_i \leq r_{\tau_i} \text{ for } i = 1, \ldots, m.$$

Define the reduced matrix $\mathbf{P}_\tau$ as the first r_τ columns of $\widehat{\mathbf{P}}_\tau$ and in the same spirit the matrices $\mathbf{P}_{\tau_i}$, for $i = 1, \ldots, m$. By this we have

$$C_\tau = \widehat{C}_\tau \times_0 \mathbf{P}_\tau^\top \bigtimes_{i=1}^{m} \mathbf{P}_{\tau_i}^*.$$

Further, let $\widetilde{\mathbf{U}}_{\tau_i}$, $i = 1, \ldots, m$ be the the matrix which contains only the first r_{τ_i} columns of $\widehat{\mathbf{U}}_{\tau_i}$, or in other words $\widetilde{\mathbf{U}}_{\tau_i} = \widehat{\mathbf{U}}_{\tau_i} \mathbf{P}_{\tau_i}$. By this we obtain

$$\widehat{X}_\tau^{\mathrm{cut}} \times_0 (\mathbf{I}_{r_\tau}, 0) = C_\tau \times_0 \mathbf{I}_{r_\tau} \bigtimes_{i=1}^{m} \widetilde{\mathbf{U}}_{\tau_i}.$$

If we set $\widetilde{X}_{\tau_i} := \widehat{X}_{\tau_i} \times_0 \mathbf{P}_{\tau_i}^\top = \widehat{X}_{\tau_i}^{\mathrm{rot}} \times_0 (\mathbf{I}_{r_\tau}, 0)$, we note the following properties

1. $\widetilde{\mathbf{U}}_{\tau_i} = \mathbf{Mat}_0(\widetilde{X}_{\tau_i})^\top$.

2. $\widetilde{X}_{\tau_i}$ only differs from $\widehat{X}^{\mathrm{rot}}_{\tau_i}$ only in that the core tensor $\widehat{C}^{\mathrm{rot}}_{\tau_i}$ is replaced by the reduced core tensor $\widetilde{C}_{\tau_i} = \widehat{C}^{\mathrm{rot}}_{\tau_i} \times_0 (\mathbf{I}_{\tau_i}, 0)$.

Recursion

The rotate and cut strategy from above is now applied recursively from the root to the leaves. At the leaves, we set $\mathbf{U}_l = \widetilde{\mathbf{U}}_l = \widehat{\mathbf{U}}_l \mathbf{P}_l$ such that all core tensors and all leaves have reduced ranks now. We set $X_{\bar{\tau}}$, consisting of all reduced leaves and core tensors, as the truncated tree tensor network. Hence, for each tree $\tau = (\tau_1, \dots, \tau_m) \le \bar{\tau}$ we have

$$X_\tau = C_\tau \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^{m} \mathbf{U}_{\tau_i} \quad \text{with } \mathbf{U}_{\tau_i} = \mathbf{Mat}_0(X_{\tau_i})^\top.$$

Since we can always find an orthonormal representation of a TTN, we can assume

$$\left\|\mathbf{Mat}_0(\widehat{C}_\tau)^\top\right\|_2 = 1, \quad \left\|\widehat{\mathbf{U}}_\tau\right\|_2 = 1 \quad \text{for all } \tau < \bar{\tau}. \tag{2.13}$$

With the considerations from above we arrive at the presented algorithm 1, but the computations are arranged in a mathematically equivalent way. This allows us to prove an error bound for the truncation algorithm depending on the tolerance parameter ϑ. The following result and its proof can be found in [CLS23, Theorem A.1].

Theorem 2.4: Rank truncation error

The error of the tree tensor network $X_{\bar{\tau}}$, which results from rank truncation of $\widehat{X}_{\bar{\tau}}$ with tolerance ϑ according to Algorithm 1, is bounded by

$$\|X_{\bar{\tau}} - \widehat{X}_{\bar{\tau}}\| \le c_{\bar{\tau}}\, \vartheta \quad \text{with} \quad c_{\bar{\tau}} = \|C_{\bar{\tau}}\|(d_{\bar{\tau}} - 1) + 1,$$

where d_τ is the number of vertices of τ.

Remark 2.1. If in algorithm 1 the tolerance at the root is chosen as $\vartheta / \|\widehat{C}_{\bar{\tau}}\|$ and kept as ϑ in the remaining truncation, the proof leads to an error bound

$$\|X_{\bar{\tau}} - \widehat{X}_{\bar{\tau}}\| \le d_{\bar{\tau}}\, \vartheta.$$

Remark 2.2. Note that the theorem 2.4 gives a linear dependence on the dimension d, while the algorithm for binary trees in [Hac12, section 11.4.2] only has a square root dependence on d.

Proof. From the sub-TTNs $\widehat{X}_\tau$ with $\tau \le \bar{\tau}$ we construct the rotated and cut TTNs $\widehat{X}^{\mathrm{rot}}_\tau$, $\widehat{X}^{\mathrm{cut}}_\tau$ as described above. For a tree $\tau = (\tau_1, \dots, \tau_m) \le \bar{\tau}$ we then have the corresponding

matrices

$$\widehat{\mathbf{U}}_\tau^{\mathrm{rot}} = \mathbf{Mat}_0(\widehat{X}_\tau^{\mathrm{rot}})^\top, \qquad \widehat{X}_\tau^{\mathrm{rot}} = \widehat{C}_\tau^{\mathrm{rot}} \times_0 \mathbf{I}_{\widehat{r}_\tau} \bigtimes_{i=1}^{m} \widehat{\mathbf{U}}_{\tau_i}^{\mathrm{rot}},$$
$$\mathbf{U}_\tau = \mathbf{Mat}_0(X_\tau)^\top, \qquad X_\tau = C_\tau \times_0 \mathbf{I}_{r_\tau} \bigtimes_{i=1}^{m} \mathbf{U}_{\tau_i}.$$

With

$$\widetilde{\mathbf{U}}_\tau = \widehat{\mathbf{U}}_\tau^{\mathrm{rot}} \begin{pmatrix} \mathbf{I}_{r_\tau} \\ 0 \end{pmatrix} = \mathbf{Mat}_0(\widetilde{X}_\tau)^\top, \qquad \widetilde{X}_\tau = \widehat{X}_\tau^{\mathrm{rot}} \times_0 (\mathbf{I}_{r_\tau}, 0),$$

we further have the intermediate tensor

$$\widetilde{X}_\tau^{\mathrm{cut}} = \widehat{X}_\tau^{\mathrm{cut}} \times_0 (\mathbf{I}_{r_\tau}, 0) = \widehat{C}_\tau^{\mathrm{cut}} \times_0 (\mathbf{I}_{r_\tau}, 0) \bigtimes_{i=1}^{m} \widehat{\mathbf{U}}_{\tau_i}^{\mathrm{rot}} = C_\tau \times_0 \mathbf{I}_{r_\tau} \bigtimes_{i=1}^{m} \widetilde{\mathbf{U}}_{\tau_i}.$$

We prove the theorem by proving the error bound

$$\|\mathbf{U}_\tau - \widetilde{\mathbf{U}}_\tau\|_2 \le d_\tau \, \vartheta \quad \text{for} \quad \tau < \bar{\tau} \tag{2.14}$$

by an induction over the height of the tree. At leaves l we have $\mathbf{U}_l = \widetilde{\mathbf{U}}_l$ and $\|\widetilde{\mathbf{U}}_l\|_2 \le 1$ by construction. As an induction hypothesis we use

$$\|\mathbf{U}_{\tau_i}\|_2 \le 1 \text{ and } \|\mathbf{U}_{\tau_i} - \widetilde{\mathbf{U}}_{\tau_i}\|_2 \le d_{\tau_i} \, \vartheta.$$

Note that $\|\widetilde{\mathbf{U}}_{\tau_i}\|_2 \le 1$ by (2.13) and by the construction of $\widetilde{\mathbf{U}}_{\tau_i}$ from $\widehat{\mathbf{U}}_{\tau_i}$. We further observe that

$$\|\mathbf{Mat}_0(C_\tau)^\top\|_2 = \|\mathbf{Mat}_0(\widehat{C}_\tau^{\mathrm{cut}})^\top\|_2 \le \|\mathbf{Mat}_0(\widehat{C}_\tau^{\mathrm{rot}})^\top\|_2 = \|\mathbf{Mat}_0(\widehat{C}_\tau)^\top\|_2$$

and

$$\|\mathbf{U}_\tau\|_2 = \|\mathbf{Mat}_0(X_\tau)^\top\|_2 = \|\bigotimes_{i=1}^{m} \mathbf{U}_{\tau_i} \, \mathbf{Mat}_0(C_\tau)^\top\|_2 \le \|\mathbf{Mat}_0(C_\tau)^\top\|_2 \prod_{i=1}^{m} \|\mathbf{U}_{\tau_i}\|_2.$$

(2.13) and the induction hypothesis yields us

$$\|\mathbf{Mat}_0(C_\tau)^\top\|_2 \le 1 \quad \text{and} \quad \|\mathbf{U}_\tau\|_2 \le 1. \tag{2.15}$$

Further, we have $\|\mathbf{U}_\tau - \widetilde{\mathbf{U}}_\tau\|_2 = \|\mathbf{Mat}_0(X_\tau - \widetilde{X}_\tau)^\top\|_2$ and write

$$X_\tau - \widetilde{X}_\tau = \left(X_\tau - \widetilde{X}_\tau^{\mathrm{cut}}\right) + \left(\widehat{X}_\tau^{\mathrm{cut}} - \widehat{X}_\tau^{\mathrm{rot}}\right) \times_0 (\mathbf{I}_{r_\tau}, 0).$$

We have a closer look a the two terms. The first one on the right-hand side equals

$$X_\tau - \widetilde{X}_\tau^{\mathrm{cut}} = C_\tau \times_0 \mathbf{I}_{r_\tau} \bigtimes_{i=1}^{m} \mathbf{U}_{\tau_i} - C_\tau \times_0 \mathbf{I}_{r_\tau} \bigtimes_{i=1}^{m} \widetilde{\mathbf{U}}_{\tau_i}$$

with the same connection tensor C_τ in both terms of the difference. We then have

$$\mathbf{Mat}_0\big(X_\tau - \widetilde{X}_\tau^{\mathrm{cut}}\big) = \mathbf{Mat}_0(C_\tau)\Big(\bigotimes_{i=1}^{m} \mathbf{U}_{\tau_i}^\top - \bigotimes_{i=1}^{m} \widetilde{\mathbf{U}}_{\tau_i}^\top\Big).$$

Writing the difference of the Kronecker products as a telescoping sum, using that $\mathbf{Mat}_0(C_\tau)^\top$ and $\mathbf{U}_{\tau_i}$ are bounded by 1 in the matrix 2-norm, and finally using the induction hypothesis, we obtain

$$\|\,\mathbf{Mat}_0\big(X_\tau - \widetilde{X}_\tau^{\mathrm{cut}}\big)^\top\|_2 \le \sum_{i=1}^{m} \|\mathbf{U}_{\tau_i} - \widetilde{\mathbf{U}}_{\tau_i}\|_2 \le \sum_{i=1}^{m} d_{\tau_i}\,\vartheta = (d_\tau - 1)\vartheta.$$

On the other hand,

$$\big(\widehat{X}_\tau^{\mathrm{cut}} - \widehat{X}_\tau^{\mathrm{rot}}\big) \times_0 (\mathbf{I}_{r_\tau}, 0) = \big(\widehat{C}_\tau^{\mathrm{cut}} - \widehat{C}_\tau^{\mathrm{rot}}\big) \times_0 (\mathbf{I}_{r_\tau}, 0) \bigtimes_{i=1}^{m} \widehat{\mathbf{U}}_{\tau_i}^{\mathrm{rot}},$$

where $\|\widehat{\mathbf{U}}_{\tau_i}^{\mathrm{rot}}\|_2 \le 1$ by construction and (2.13). In total we have

$$\begin{aligned}
&\|\,\mathbf{Mat}_0\big((\widehat{X}_\tau^{\mathrm{cut}} - \widehat{X}_\tau^{\mathrm{rot}}) \times_0 (\mathbf{I}_{r_\tau}, 0)\big)^\top\|_2 \le \|\,\mathbf{Mat}_0\big(\widehat{C}_\tau^{\mathrm{cut}} - \widehat{C}_\tau^{\mathrm{rot}}\big)^\top\|_2 \\
&\le \|\,\mathbf{Mat}_0\big(\widehat{C}_\tau^{\mathrm{cut}}\big) - \mathbf{Mat}_0\big(\widehat{C}_\tau^{\mathrm{rot}}\big)\|_F = \|\,\mathbf{Mat}_1\big(\widehat{C}_\tau^{\mathrm{cut}}\big) - \mathbf{Mat}_1\big(\widehat{C}_\tau^{\mathrm{rot}}\big)\|_F \le \vartheta,
\end{aligned}$$

where we used (2.10)–(2.12) in the last inequality. Altogether we find

$$\begin{aligned}
\|\mathbf{U}_\tau - \widetilde{\mathbf{U}}_\tau\|_2 &\le \|\,\mathbf{Mat}_0\big(X_\tau - \widetilde{X}_\tau^{\mathrm{cut}}\big)^\top\|_2 + \|\,\mathbf{Mat}_0\big((\widehat{X}_\tau^{\mathrm{cut}} - \widehat{X}_\tau^{\mathrm{rot}}) \times_0 (\mathbf{I}_{r_\tau}, 0)\big)^\top\|_2 \\
&\le d_\tau\,\vartheta,
\end{aligned}$$

which completes the proof of (2.14) by induction. Finally, for the full tree $\bar\tau = (\tau_1, \dots, \tau_m)$, where $\widehat{r}_{\bar\tau} = r_{\bar\tau} = 1$ but $\|C_{\bar\tau}\|$ is arbitrary, we use the same argument as above in estimating the norm of

$$X_{\bar\tau} - \widehat{X}_{\bar\tau} = X_{\bar\tau} - \widehat{X}_{\bar\tau}^{\mathrm{rot}} = \Big(C_{\bar\tau} \bigtimes_{i=1}^{m} \mathbf{U}_{\tau_i} - C_{\bar\tau} \bigtimes_{i=1}^{m} \widetilde{\mathbf{U}}_{\tau_i}\Big) + \big(\widehat{C}_{\bar\tau}^{\mathrm{cut}} - \widehat{C}_{\bar\tau}^{\mathrm{rot}}\big) \bigtimes_{i=1}^{m} \widehat{\mathbf{U}}_{\tau_i}^{\mathrm{rot}}.$$

This yields $\|X_{\bar\tau} - \widehat{X}_{\bar\tau}\| \le \big(\|C_{\bar\tau}\|\,(d_{\bar\tau} - 1) + 1\big)\vartheta$. □

2.5 Geometry of Tree Tensor Networks

In the course of the thesis, we will see that the framework of dynamical low-rank approximation is based on the Dirac-Frenkel time-dependent variational principle, see section 2.6. This involves a projection onto the tangent space at the current approximation point on a manifold. Hence, before going into the details of time integration, it is worth having a closer look into the differential geometry needed to describe the set of tree tensor networks. In the following, we recap differential geometric fundamentals based on [Bou23].

To do so we first define smooth embedded submanifolds of a linear space. The definition is taken from [Bou23, Definition 3.10].

Definition 2.5: Smooth embedded submanifold

Let $\mathcal{E}$ be a linear space of dimension d. A non-empty subset $\mathcal{M}$ of $\mathcal{E}$ is a (smooth) embedded submanifold of $\mathcal{E}$ of dimension n if either

1. $n = d$ and $\mathcal{M}$ is open in $\mathcal{E}$. We call this an open submanifold.
2. $n = d - k$ for some $k \geq 1$ and for each $x \in \mathcal{M}$, there exists a neighborhood U of x in $\mathcal{E}$ and a smooth function $h : U \to \mathbb{R}^k$ such that
   (a) If $y \in U$, then $h(y) = 0$ if and only if $y \in \mathcal{M}$.
   (b) $\operatorname{rank}(Dh(x)) = k$.

   Here, Dh denotes the Jacobian of h. The function h is called a local defining function for $\mathcal{M}$ at x.

The definition can be interpreted in the way that a smooth set $\mathcal{M}$ can be approximated in some meaningful way around each point $x \in \mathcal{M}$. This linearisation around a point $x \in \mathcal{M}$ will be called the tangent space. The following definition can be found in [Bou23, Definition 3.14 and 3.16].

Definition 2.6: Tangent space

Let $\mathcal{M}$ be a subset of $\mathcal{E}$. For all $x \in \mathcal{M}$, define

$$\mathcal{T}_x\mathcal{M} = \left\{ c'(0) \mid c : I \to \mathcal{M} \text{ is smooth and } c(0) = x \right\},$$

where I is any open interval containing $t = 0$. That is, v is in $\mathcal{T}_x\mathcal{M}$ if and only if there exists a smooth curve on $\mathcal{M}$ passing through x with velocity v. We call $\mathcal{T}_x\mathcal{M}$ the tangent space to $\mathcal{M}$ at x.

A different characterization of the tangent space is often useful. It can be described as the kernel of the derivative of the corresponding local defining function at point x. More precisely, we have the following theorem.

Theorem 2.5: Tangent space representation

Let $\mathcal{M}$ be a smooth embedded submanifold with dimension $k < n$ and $h : U \to \mathbb{R}^k$ the local defining function for $\mathcal{M}$ at x. Then the tangent space at $x \in \mathcal{M}$ can be represented by

$$\mathcal{T}_x\mathcal{M} = \ker Dh(x),$$

where Dh denotes the Jacobi matrix of h.

The theorem is taken from [Bou23, Theorem 3.15], where also a proof can be found.

2.5.1 Geometry of matrices

Orthonormal matrices

Fix a rank r and consider the complex inner product $\langle u, v\rangle = u^* v$ for $u, v \in \mathbb{C}^n$. For $n > r$ we define set of matrices $\mathbf{U} \in \mathbb{C}^{n\times r}$ whose columns are orthonormal by

$$\mathcal{V}_{n,r} := \{\mathbf{U} \in \mathbb{C}^{n\times r} : \mathbf{U}^*\mathbf{U} = \mathbf{I}_r\},$$

where $\mathbf{I}_r$ is the identity matrix of size $r \times r$. For real matrices and replacing the $*$ operation with the regular transpose, it is shown in [Bou23, Section 7.3] that $\mathcal{V}_{n,r}$ is an embedded submanifold of $\mathbb{R}^{n\times r}$. This manifold is called *Stiefel manifold*. To prove that it is an embedded submanifold, we follow the same strategy. The local defining function for the Stiefel manifold equals

$$h : \mathbb{C}^{n\times r} \to \mathrm{Sym}(r) \text{ with } \mathbf{U} \mapsto \mathbf{U}^*\mathbf{U} - \mathbf{I}_r,$$

where $\mathrm{Sym}(r)$ is the linear space of unitary matrices of size r, see the real case in [Bou23]. This allows us to characterize the tangent space in the following way. Deriving the local defining function we obtain

$$Dh(\boldsymbol{X})[\mathbf{U}] = \boldsymbol{X}^*\mathbf{U} + \mathbf{U}^*\boldsymbol{X}.$$

Hence, the tangent space at $\boldsymbol{X} \in \mathcal{V}_{n,r}$ equals

$$\begin{aligned}\mathcal{T}_{\boldsymbol{X}}\mathcal{V}_{n,r} &= \{\mathbf{U} \in \mathbb{C}^{n\times r} : \boldsymbol{X}^*\mathbf{U} + \mathbf{U}^*\boldsymbol{X} = 0\} \\ &= \{\mathbf{U} \in \mathbb{C}^{n\times r} : \boldsymbol{X}^*\mathbf{U} \in \mathrm{Skew}(r)\},\end{aligned} \tag{2.16}$$

where $\mathrm{Skew}(r)$ denotes the space of skew-symmetric complex $r \times r$ matrices.

Matrices of fixed rank r

We now consider the set of matrices of size $n \times m$ of a fixed rank r,

$$\mathcal{M}_r = \{\boldsymbol{A} \in \mathbb{C}^{n\times m} : \mathrm{rank}(\boldsymbol{A}) = r\}. \tag{2.17}$$

Typically we are interested in situations where $r \ll n, m$. Again we want to show that $\mathcal{M}_r$ indeed is an embedded submanifold, for which we follow the ideas of [Bou23, Section 7.5]. Let $\mathcal{U}$ be the subset of $\mathbb{C}^{n\times m}$, where the upper left block of size $r \times r$ is invertible. Then consider the function

$$h : \mathcal{U} \to \mathbb{C}^{(n-r)\times(m-r)} : \boldsymbol{Y} = \begin{pmatrix} \boldsymbol{Y}_{11} & \boldsymbol{Y}_{12} \\ \boldsymbol{Y}_{21} & \boldsymbol{Y}_{22} \end{pmatrix} \mapsto h(\boldsymbol{Y}) = \boldsymbol{Y}_{22} - \boldsymbol{Y}_{21}\boldsymbol{Y}_{11}^{-1}\boldsymbol{Y}_{12}.$$

In [Bou23, Section 7.5] it is shown that h is indeed a local defining function for $\mathcal{M}_r$ and hence $\mathcal{M}_r$ an embedded submanifold.

Remark 2.3. Note that $\mathcal{M}_r$ is an embedded submanifold only for fixed rank r. If we allow for matrices of rank up to r, the obtained set is no longer an embedded submanifold. However, it is an algebraic variety [Bou23, Section 7.5].

We remind that each element of $\mathcal{M}_r$ can be decomposed by a singular value decomposition of the form

$$\boldsymbol{Y} = \mathbf{U}\mathbf{S}\mathbf{V}^*, \text{ with } \mathbf{U} \in \mathbb{C}^{n\times r}, \mathbf{V} \in \mathbb{C}^{m\times r}, \text{ unitary and } \mathbf{S} \in \mathbb{C}^{r\times r},$$

see section 2.1. By this, it is shown in [Bou23] that the elements of the tangent space $\mathcal{T}_{\boldsymbol{Y}}\mathcal{M}_r$ have the form

$$\begin{aligned}\mathcal{T}_{\boldsymbol{Y}}\mathcal{M}_r = \{\mathbf{U}\boldsymbol{M}\mathbf{V}^* + \delta\mathbf{U}\mathbf{V}^* + \mathbf{U}\delta\mathbf{V}^* : \boldsymbol{M} \in \mathbb{C}^{r\times r}, \delta\mathbf{U} \in \mathbb{C}^{n\times r}, \delta\mathbf{V} \in \mathbb{C}^{m\times r}, \\ \text{and } \mathbf{U}^*\delta\mathbf{U} = 0 = \mathbf{V}^*\delta\mathbf{V}\}.\end{aligned} \tag{2.18}$$

Although we now have a characterization of the tangent elements, it will be more convenient for us to use a different representation. As in [KL07], we consider the tangent

map of $(\mathbf{S}, \mathbf{U}, \mathbf{V}) \mapsto \boldsymbol{Y} = \mathbf{U}\mathbf{S}\mathbf{V}^*$,

$$\begin{aligned}\mathbb{C}^{r\times r} \times \mathcal{T}_{\mathbf{U}}\mathcal{V}_{n,r} \times \mathcal{T}_{\mathbf{V}}\mathcal{V}_{m,r} &\to \mathcal{T}_{\boldsymbol{Y}}\mathcal{M}_r \times \mathrm{Skew}(r) \times \mathrm{Skew}(r)\\ (\delta\mathbf{S}, \delta\mathbf{U}, \delta\mathbf{V}) &\mapsto (\delta\mathbf{U}\mathbf{S}\mathbf{V}^* + \mathbf{U}\delta\mathbf{S}\mathbf{V}^* + \mathbf{U}\mathbf{S}\delta\mathbf{V}^*, \mathbf{U}^*\delta\mathbf{U}, \mathbf{V}^*\delta\mathbf{V}).\end{aligned}$$

Clearly this is a linear map with trivial kernel. It further can be verified that the dimensions of the vector spaces coincide. Thus, the map above is an isomorphism, which implies that the tangent space of $\mathcal{M}_r$ can be also represented by

$$\mathcal{T}_{\boldsymbol{Y}}\mathcal{M}_r = \{\delta\mathbf{U}\mathbf{S}\mathbf{V}^* + \mathbf{U}\delta\mathbf{S}\mathbf{V}^* + \mathbf{U}\mathbf{S}\delta\mathbf{V}^* : \delta\mathbf{S} \in \mathbb{C}^{r\times r}, \delta\mathbf{U} \in \mathcal{T}_U\mathcal{V}_{n,r}, \delta\mathbf{V} \in \mathcal{T}_V\mathcal{V}_{m,r}\}. \quad (2.19)$$

Further, the factors $\delta\mathbf{U}, \delta\mathbf{S}, \delta\mathbf{V}$ are uniquely determined if we impose the orthogonality constraints

$$\mathbf{U}^*\delta\mathbf{U} = 0, \quad \mathbf{V}^*\delta\mathbf{V} = 0.$$

It remains to study the orthogonal projection onto the tangent space at point $\boldsymbol{Y}$ denoted by $P(\boldsymbol{Y})$. For this, we remark [KL07, Lemma 4.1]:

Lemma 2.2: Tangent space projection for matrices

The orthogonal projection onto the tangent space $\mathcal{T}_{\boldsymbol{Y}}\mathcal{M}_r$ at the point $\boldsymbol{Y} = \mathbf{U}\mathbf{S}\mathbf{V}^* \in \mathcal{M}_r$ is given by

$$P(\boldsymbol{Y}) = \mathbf{I} - P^{\perp}(\boldsymbol{Y}), \text{ with } P^{\perp}(\boldsymbol{Y})\boldsymbol{Z} = (\mathbf{I} - \mathbf{U}\mathbf{U}^*)\boldsymbol{Z}(\mathbf{I} - \mathbf{V}\mathbf{V}^*).$$

Applying the projection to an element $\boldsymbol{Z}$ then yields

$$\begin{aligned}P(\boldsymbol{Y})\boldsymbol{Z} &= \boldsymbol{Z} - (\mathbf{I} - \mathbf{U}\mathbf{U}^*)\boldsymbol{Z}(\mathbf{I} - \mathbf{V}\mathbf{V}^*)\\ &= \mathbf{U}\mathbf{U}^*\boldsymbol{Z}\mathbf{V}\mathbf{V}^* + \mathbf{U}\mathbf{U}^*\boldsymbol{Z}(\mathbf{I} - \mathbf{V}\mathbf{V}^*) + (\mathbf{I} - \mathbf{U}\mathbf{U}^*)\boldsymbol{Z}\mathbf{V}\mathbf{V}^*. \quad (2.20)\end{aligned}$$

2.5.2 Geometry of Tucker tensors

For a given multilinear rank $r = (r_1, \ldots, r_d)$, we consider the set of tensors

$$\mathcal{M}_r = \{Y \in \mathbb{C}^{n_1\times\cdots\times n_d} : Y \text{ has multilinear rank } r\},$$

which is indeed a manifold [KL10]. It will serve as the approximation manifold, i.e. general tensors $A \in \mathbb{C}^{n_1\times\cdots\times n_d}$ will be approximated by an element in $\mathcal{M}_r$. As described

in detail in section 2.2, each element of $\mathcal{M}_r$ can be decomposed into a Tucker decomposition, such that we have a memory-efficient approximation of a tensor, i.e.

$$Y = C \bigtimes_{i=1}^{d} \mathbf{U}_i.$$

Similar to the matrix case and following [KL10], we consider the extended tangent map $(C, \mathbf{U}_1, \dots, \mathbf{U}_d) \mapsto Y = C \times_{i=1}^{d} \mathbf{U}_i$

$$\begin{aligned} \mathbb{C}^{r_1 \times \cdots \times r_d} \times \prod_{i=1}^{d} \mathcal{T}_{\mathbf{U}_i} \mathcal{V}_{n_i, r_i} &\to \mathcal{T}_Y \mathcal{M}_r \times \prod_{i=1}^{d} \operatorname{Skew}(r_i) \\ (\delta C, \delta \mathbf{U}_1, \dots, \delta \mathbf{U}_d) &\mapsto \left(\delta C \bigtimes_{i=1}^{d} \mathbf{U}_i + \sum_{i=1}^{d} C \times_i \delta \mathbf{U}_i \bigtimes_{j \neq i} \mathbf{U}_j, \mathbf{U}_1^* \delta \mathbf{U}_1, \dots, \mathbf{U}_d^* \delta \mathbf{U}_d \right). \end{aligned}$$

With the same arguments as for the matrix case, we obtain that the tangent space at $Y = C \times_{i=1}^{d} \mathbf{U}_i$ can be represented by

$$\mathcal{T}_Y \mathcal{M}_r = \left\{ \delta C \bigtimes_{i=1}^{d} \mathbf{U}_i + \sum_{i=1}^{d} C \times_i \delta \mathbf{U}_i \bigtimes_{j \neq i} \mathbf{U}_j : \delta C \in \mathbb{C}^{r_1 \times \cdots \times r_d}, \delta \mathbf{U}_i \in \mathcal{T}_{\mathbf{U}_i} \mathcal{V}_{n_i, r_i} \right\}. \tag{2.21}$$

The factors $\delta C, \delta \mathbf{U}_1, \dots, \delta \mathbf{U}_d$ are uniquely determined if we impose an orthogonality constraint

$$\mathbf{U}_i^* \delta \mathbf{U}_i = 0 \quad \forall i = 1, \dots, d. \tag{2.22}$$

The tangent space projection from lemma 2.2 can be generalized to Tucker tensors, see [KL10, Lemma 3.1].

Lemma 2.3: Tangent space projection Tucker tensors

The orthogonal projection onto the tangent space $\mathcal{T}_Y \mathcal{M}_r$ at the point $Y = C \times_{i=1}^{d} \mathbf{U}_i \in \mathcal{M}_r$ is given by

$$P(Y)Z = Z \bigtimes_{i=1}^{d} \mathbf{U}_i \mathbf{U}_i^* + \sum_{i=1}^{d} Z \times_i (\mathbf{I} - \mathbf{U}_i \mathbf{U}_i^*) \bigtimes_{j \neq i} \mathbf{U}_j \mathbf{U}_j^*.$$

2.5.3 Geometry of hierarchical Tucker tensors

Fix a binary tree $\bar{\tau}$ with d leaves and also fix the tree ranks $r = (r_\tau)_{\tau \leq \bar{\tau}}$. Then consider the set of hierarchical Tucker tensors with fixed tree ranks r

$$\mathcal{M}_r = \{ Y \in \mathbb{C}^{n_1 \times \cdots \times n_d} \text{ in HT representation} : \ Y \text{ has tree ranks } r = (r_\tau)_{\tau \leq \bar{\tau}} \}.$$

In [UV13] it is shown for real tensors that $\mathcal{M}_r$ is an embedded manifold of $\mathbb{R}^{n_1 \times \cdots \times n_d}$. If the tensor $Y \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ is considered in the tensor train format, it was shown in [HRS12] that the set of tensors in tensor train format with fixed tree ranks is also an embedded manifold. To the author's knowledge, there is no literature on the geometry of general tree tensor networks so far. Since the geometry is analysed for the real hierarchical Tucker and tensor train format, most likely similar results can be obtained in the tree tensor network setting with complex numbers. However, this lies beyond the scope of this thesis and could be investigated in the future.

In the course of this thesis, we will consider the manifold of tensors in tree tensor network format for a given tree $\bar{\tau}$, dimensions $(n_l)_{l=1}^d$ and tree ranks $r = (r_\tau)_{\tau \leq \bar{\tau}}$. By $\mathcal{M}_{\bar{\tau}} = \mathcal{M}_{\bar{\tau},r} = \mathcal{M}(\bar{\tau}, (n_l)_{l=1}^d, (r_\tau)_{\tau \leq \bar{\tau}})$ we denote the corresponding embedded manifold in the tensor space $\mathbb{C}^{n_1 \times \cdots \times n_d}$.

2.6 The Dirac-Frenkel time-dependent variational principle

Consider a abstract Hilbert space $\mathcal{H}$ together with an inner product $\langle \cdot | \cdot \rangle$. On this Hilbert space we consider a tensor differential equation of the same form as in equation (2.1)

$$\partial_t u(t) = F(t, u(t)), \tag{2.23}$$

for a general right-hand side F. The following notation is taken from [Lub08]. To derive the Dirac-Frenkel time-dependent variational principle we take a smooth submanifold $\mathcal{M}$ of $\mathcal{H}$. Further let $\mathcal{T}_u\mathcal{M}$ be the tangent space at the point u. The submanifold $\mathcal{M}$ will be the approximation space on which we want to find an approximation $u(t)$ to actual solution.
Following [Lub08], we determine the approximation $u(t)$ from the condition that at time t its derivative $\partial_t u(t)$, which lies in $\mathcal{T}_{u(t)}\mathcal{M}$, fulfills

$$\partial_t u(t) \in \mathcal{T}_{u(t)}\mathcal{M} \quad \text{such that} \quad \langle v | \partial_t u - F(t, u(t)) \rangle = 0 \ \forall v \in \mathcal{T}_{u(t)}\mathcal{M}\,. \tag{2.24}$$

Taking the real part of (2.24) we obtain a minimal condition of the form

$$\partial_t u \text{ is chosen as the } w \in \mathcal{T}_u\mathcal{M} \text{ for which } \|w - F(t, u(t))\| \text{ is minimal,}$$

see again [Lub08]. Hence, $\partial_t u$ can be interpreted as an orthogonal projection of $F(t, u(t))$ onto $\mathcal{T}_u\mathcal{M}$. Graphically this is illustrated in figure 2.7.

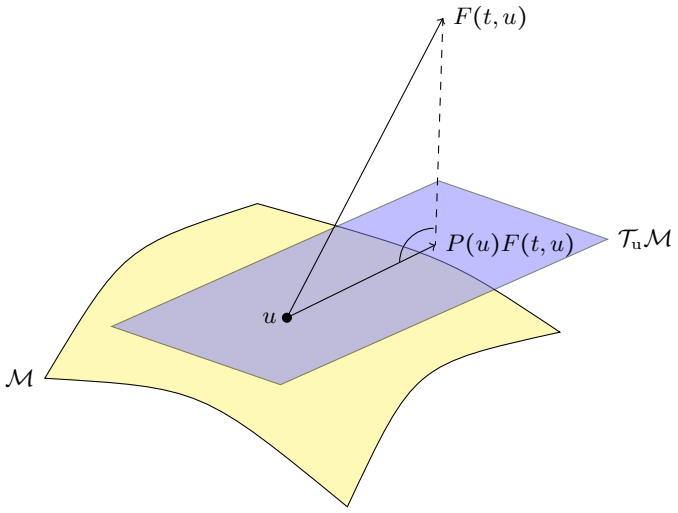

FIGURE 2.7: Illustration of the Dirac-Frenkel time-dependent variational principle.

By denoting the orthogonal projection onto $\mathcal{T}_u\mathcal{M}$, we can rewrite the Driac-Frenkel time-dependent variational principle (2.24) as a projected differential equation on the manifold $\mathcal{M}$, i.e.

$$\partial_t u(t) = P(u)F(t,u(t)). \tag{2.25}$$

Already in 1930, Dirac used the variational formulation (2.24) with F being a Hamiltonian as a right-hand side to derive the equations of motion of the time-dependent Hartree–Fock method [Dir30]. In 1934 Frenkel gave the interpretation as an orthogonal projection, see [Fre34]. This is why the variational principle is called the Dirac-Frenkel variational principle.

Application to matrix differential equations

For simplification reasons, we first stick to matrix differential equations, i.e. $d = 2$ in (2.23). We choose $\mathcal{M}_r$ to be the manifold of matrices of rank r. By a svd-like decomposition, we can write the approximation $\boldsymbol{u}(t)$ in a time-dependent decomposition as

$$\boldsymbol{u}(t) = \mathbf{U}(t)\mathbf{S}(t)\mathbf{V}(t)^*, \text{ with } \quad \mathbf{U}(t) \in \mathbb{C}^{n\times r},\ \mathbf{S}(t) \in \mathbb{C}^{r\times r},\ \mathbf{V}(t) \in \mathbb{C}^{m\times r},$$

where $\mathbf{U}(t)$ and $\mathbf{V}(t)$ are orthogonal matrices for all times t. Additionally, we require that $\mathbf{S}(t)$ is invertible for all times t. To uniquely determine the elements $\partial_t \boldsymbol{u}$ in the tangent space $\mathcal{T}_u\mathcal{M}$, we require $\mathbf{U}^*\dot{\mathbf{U}} = 0 = \mathbf{V}^*\dot{\mathbf{V}}$ along the solution trajectory. By (2.19)

we know now that we can write $\partial_t \boldsymbol{u}$ as

$$\partial_t \boldsymbol{u} = \dot{\mathbf{U}}\mathbf{S}\mathbf{V}^* + \mathbf{U}\dot{\mathbf{S}}\mathbf{V}^* + \mathbf{U}\mathbf{S}\dot{\mathbf{V}}^*,$$

where $\dot{\mathbf{S}} \in \mathbb{C}^{r\times r}$, $\dot{\mathbf{U}} \in \mathcal{T}_U\mathcal{V}_{n,r}$ and $\dot{\mathbf{V}} \in \mathcal{T}_V\mathcal{V}_{m,r}$. We obtain equations of motion for the factors $\mathbf{U}(t), \mathbf{V}(t)$ and $\mathbf{S}(t)$ by using the Dirac-Frenkel variational principle. First, we derive the equation of motion for $\mathbf{S}(t)$. As an element of the tangent space $\mathcal{T}_{\mathrm{u}}\mathcal{M}$, we choose $v = u_i v_j^*$, where u_i and v_j are the ith column of $\mathbf{U}$ and the jth column of $\mathbf{V}$ respectively. Further, we make use of the identity

$$\langle vw^* | \boldsymbol{B} \rangle = v^* \boldsymbol{B} w,$$

for a matrix $\boldsymbol{B} \in \mathbb{C}^{n\times m}$ and vectors $v \in \mathbb{C}^n$ and $w \in \mathbb{C}^m$. Inserting now $v = u_i v_j^*$, we obtain

$$\begin{aligned}
0 &= \langle u_i v_j^* | \partial_t \boldsymbol{u} - \boldsymbol{F}(t, \boldsymbol{u}(t)) \rangle \\
&= \langle u_i v_j^* | \dot{\mathbf{U}}\mathbf{S}\mathbf{V}^* + \mathbf{U}\dot{\mathbf{S}}\mathbf{V}^* + \mathbf{U}\mathbf{S}\dot{\mathbf{V}}^* - \boldsymbol{F}(t, \boldsymbol{u}(t)) \rangle \\
&= u_i^* \dot{\mathbf{U}}\mathbf{S}\mathbf{V}^* v_j + u_i^* \mathbf{U}\dot{\mathbf{S}}\mathbf{V}^* v_j + u_i^* \mathbf{U}\mathbf{S}\dot{\mathbf{V}}^* v_j - u_i^* \boldsymbol{F}(t, \boldsymbol{u}(t)) v_j \\
&= \dot{S}_{ij} - u_i^* \boldsymbol{F}(t, \boldsymbol{u}(t)) v_j,
\end{aligned}$$

where we used the Gauge conditions $\mathbf{U}^*\dot{\mathbf{U}} = 0 = \mathbf{V}^*\dot{\mathbf{V}}$ and $\dot{S}_{ij}$ denotes the (i,j) entry of $\dot{\mathbf{S}}$. Thus, we have the equation of motion for $\mathbf{S}(t)$ by

$$\dot{\mathbf{S}}(t) = \mathbf{U}^* \boldsymbol{F}(t, \boldsymbol{u}(t)) \mathbf{V}.$$

Analogously, one obtains the equations of motion for the factors $\mathbf{U}(t)$ and $\mathbf{V}(t)$ by inserting $v = \delta U S_{ij} v_j^*$ and $v = u_j S_{ji} \delta V$ respectively into (2.24), where $\delta U \in \mathbb{C}^n$, $\delta V \in \mathbb{C}^m$ and $\delta U^* \mathbf{U} = 0 = \delta V^* \mathbf{V}$. Together, we arrive to the following equations of motion for the low-rank factors, cf. [KL07],

$$\begin{aligned}
\dot{\mathbf{S}}(t) &= \mathbf{U}(t)^* \boldsymbol{F}(t, \boldsymbol{u}(t)) \mathbf{V}(t) \\
\dot{\mathbf{U}}(t) &= (\mathbf{I} - \mathbf{U}(t)\mathbf{U}(t)^*) \boldsymbol{F}(t, \boldsymbol{u}(t)) \mathbf{V}(t) \mathbf{S}(t)^{-1} \\
\dot{\mathbf{V}}(t) &= (\mathbf{I} - \mathbf{V}(t)\mathbf{V}(t)^*) \boldsymbol{F}(t, \boldsymbol{u}(t))^* \mathbf{U}(t) \mathbf{S}(t)^{-*}.
\end{aligned}$$

The inverse of the $\mathbf{S}(t)$ matrix for the equation for $\mathbf{U}(t)$ and $\mathbf{V}(t)$ leads to numerical instabilities, when singular values of $\mathbf{S}(t)$ become small. Note that the derivation of the equations of motion has also been done for Tucker tensors in [KL10] and for the hierarchical Tucker format and tensor train format in [LRSV13].

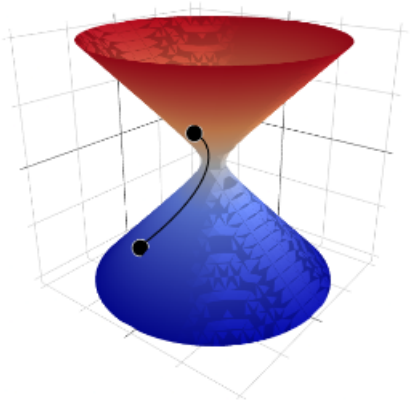

FIGURE 2.8: Trajectory of $\mathbf{U}(t)\mathbf{S}(t)\mathbf{V}(t)^*$ in the presence of high curvature of the manifold.

The inverse of $\mathbf{S}(t)$ may cause problems in numerical computations if $\mathbf{S}(t)$ is not invertible or close to singular. This is equivalent to the presence of small singular values. Let σ_r be the smallest singular value of $\mathbf{S}(t)$ for a fixed time t. In [KL07, Lemma 4.1] it was shown that for $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{M}_r$ with $\sigma_r(\boldsymbol{X}) \geq \rho > 0$ and $\|\boldsymbol{Y} - \boldsymbol{X}\| \leq \frac{1}{8}\rho$ it holds

$$\|(P(\boldsymbol{Y}) - P(\boldsymbol{X}))\boldsymbol{B}\| \leq 8\rho^{-1}\|\boldsymbol{Y} - \boldsymbol{X}\|\|\boldsymbol{B}\|_2,$$

for all $\boldsymbol{B} \in \mathbb{R}^{n\times m}$. Thus, the projection onto the tangent space is a function with a high Lipschitz constant if $\mathbf{S}(t)$ contains small singular values. Consequently stiffness in the projected differential equation (2.25) goes with the small singular values in $\mathbf{S}(t)$. In this situation, one needs very small time step sizes to solve the equations accurately. Further, there is an interesting connection between small singular values and the curvature of the manifold. In [FL18, Theorem 24] the authors show that the maximal curvature of the manifold $\mathcal{M}_r$ is σ_r^{-1}, which diverges if σ_r goes to zero. Moreover, $\mathcal{M}_r$ can be viewed as a collection of cones or as a multidimensional spiral. A possible trajectory for the equations of motion in the situation of small singular values is illustrated in figure 2.8.

Stable time integration of factors

In consideration of small singular values one would like to derive equations of motion, which are not affected by small singular values. In the following, we briefly discuss one possible way how to derive a stable numerical scheme to solve the projected ODE (2.25).

We start by multiplying the equations for $\mathbf{U}(t)$ and $\mathbf{V}(t)$ with $\mathbf{S}(t)$ and $\mathbf{S}(t)^*$ respectively, from the right, which leads to

$$\dot{\mathbf{S}}(t) = \mathbf{U}(t)^* \boldsymbol{F}(t, \boldsymbol{u}(t))\mathbf{V}(t) \tag{2.26}$$

$$\dot{\mathbf{U}}(t)\mathbf{S}(t) = (\mathbf{I} - \mathbf{U}(t)\mathbf{U}(t)^*)\boldsymbol{F}(t, \boldsymbol{u}(t))\mathbf{V}(t) \tag{2.27}$$

$$\dot{\mathbf{V}}(t)\mathbf{S}(t)^\top = (\mathbf{I} - \mathbf{V}(t)\mathbf{V}(t)^*)\boldsymbol{F}(t, \boldsymbol{u}(t))^*\mathbf{U}(t). \tag{2.28}$$

We introduce now the auxiliary variables $\mathbf{K}(t) = \mathbf{U}(t)\mathbf{S}(t)$ and $\mathbf{L}(t) = \mathbf{V}(t)\mathbf{S}(t)^*$. By the chain rule and inserting the equations of motion for $\dot{\mathbf{U}}(t)$ and $\dot{\mathbf{S}}(t)$ we obtain

$$\begin{aligned}\dot{\mathbf{K}}(t) &= \dot{\mathbf{U}}(t)\mathbf{S}(t) + \mathbf{U}(t)\dot{\mathbf{S}}(t) \\ &= (\mathbf{I} - \mathbf{U}(t)\mathbf{U}(t)^*)\boldsymbol{F}(t, \boldsymbol{u}(t))\mathbf{V}(t)\mathbf{S}(t)^{-1}\mathbf{S}(t) + \mathbf{U}(t)\mathbf{U}(t)^*\boldsymbol{F}(t, \boldsymbol{u}(t))\mathbf{V}(t) \\ &= \boldsymbol{F}(t, \boldsymbol{u}(t))\mathbf{V}(t) = \boldsymbol{F}(t, \mathbf{U}(t)\mathbf{S}(t)\mathbf{V}(t)^*)\mathbf{V}(t). \end{aligned} \tag{2.29}$$

Analogously, we obtain for $\dot{\mathbf{L}}$

$$\dot{\mathbf{L}}(t) = \boldsymbol{F}(t, \boldsymbol{u}(t))^*\mathbf{U}(t) = \boldsymbol{F}(t, \mathbf{U}(t)\mathbf{S}(t)\mathbf{V}(t)^*)^*\mathbf{U}(t). \tag{2.30}$$

Clearly, solving the $\mathbf{K}$ and $\mathbf{L}$ equations (2.29) and (2.30) is still equivalent to solving the equations of motion from (2.27) and (2.28).

Finally, we want to decouple the system of ODEs. To do that in the $\dot{\mathbf{K}}$ equation, we introduce the approximation that over one time step from t_0 to t_1, we keep the matrix $\mathbf{V}$ fixed. In an abuse of notation, we reuse $\mathbf{K}$ (and later as well $\mathbf{L}$) and solve the the ODE

$$\begin{aligned}\dot{\mathbf{K}}(t) &= \boldsymbol{F}(t, \mathbf{U}(t)\mathbf{S}(t)\mathbf{V}(t_0)^*)\mathbf{V}(t_0) \\ &= \boldsymbol{F}(t, \mathbf{K}(t)\mathbf{V}(t_0)^*)\mathbf{V}(t_0) \text{ with } \mathbf{K}(t_0) = \mathbf{U}(t_0)\mathbf{S}(t_0)\end{aligned}$$

from t_0 to t_1. Similarly, to approximate the solution of $\dot{\mathbf{L}}$, we keep $\mathbf{U}(t)$ fixed and solve the ODE

$$\begin{aligned}\dot{\mathbf{L}}(t) &= \boldsymbol{F}(t, \mathbf{U}(t_0)\mathbf{S}(t)\mathbf{V}(t)^*)^*\mathbf{U}(t_0) \\ &= \boldsymbol{F}(t, \mathbf{U}(t_0)\mathbf{L}(t)^*)^*\mathbf{U}(t_0), \text{ with } \mathbf{L}(t_0) = \mathbf{V}(t_0)\mathbf{S}(t_0)^*\end{aligned}$$

form t_0 to t_1. For the $\dot{\mathbf{S}}$ equation we simply perform a Galerkin ansatz in the new basis $\mathbf{U}(t_1)$ and $\mathbf{V}(t_1)$. Therefore we solve from t_0 to t_1

$$\begin{aligned}\dot{\tilde{\mathbf{S}}}(t) &= \mathbf{U}(t_1)^*\boldsymbol{F}(t, \mathbf{U}(t_1)\tilde{\mathbf{S}}(t)\mathbf{V}(t_1)^*)\mathbf{V}(t_1), \\ \tilde{\mathbf{S}}(t_0) &= \mathbf{U}(t_1)^*\mathbf{U}(t_0)\mathbf{S}(t_0)\mathbf{V}(t_0)^*\mathbf{V}(t_1).\end{aligned}$$

Note that if the time step size goes to zero, we are still solving the projected matrix ODE (2.25) exactly.

The three differential equations are the ones of the fixed-rank BUG integrator for matrices [CL21], which is also called the unconventional integrator. In the same work, an error bound was proven where all appearing constants are completely independent of the singular values. Thus, the presented approximation leads to a numerical stable time integration scheme. Solving these different equations of motion can be geometrically interpreted by moving along flat subspaces within the manifold. As derived by [FL18], the manifold consists of cones and therefore the existence of straight lines in $\mathcal{M}_r$ is guaranteed. The differential equation for $\dot{\mathbf{K}}$ follows a straight line, such that the high curvature in the manifold is not seen. Analogous results hold for $\dot{\mathbf{L}}$ and $\dot{\mathbf{S}}$. The movement along flat subspaces is illustrated for the $\mathbf{K}$ step in figure 2.9.

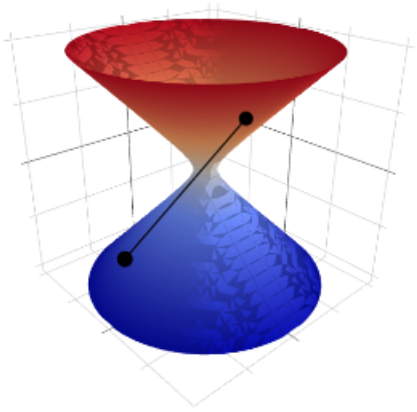

FIGURE 2.9: Trajectory of $\mathbf{U}(t)\mathbf{S}(t)\mathbf{V}(t_0)^*$ for the $\mathbf{K}$-step. The high curvature is not seen along this trajectory.

The remainder of this thesis is dedicated to the construction and analysis of such numerical stable time integration methods for tree tensor networks. Further tools, which are needed for efficient computations in quantum dynamics, are discussed to obtain fast and robust time integration.

Chapter 3

Tree Tensor Network Operators

This chapter is mainly based on the work "Low-Rank Tree Tensor Network Operators for Long-Range Pairwise Interactions" by Gianluca Ceruti, Daniel Kressner and the author [CKS24].

Tensor methods have demonstrated their efficacy in addressing high-dimensional problems and the appearing curse of dimensionality. The effectiveness of these methods rely fundamentally on the representation of a suitable representation of the operator describing the system. In this chapter, we introduce Tree Tensor Network Operators (TTNOs). We will describe how an operator in tree tensor network format can be applied efficiently to a TTN. Further, we give an explicit construction of a TTNO for long-range pairwise interacting quantum spin systems of the form

$$\mathcal{H} : \mathbb{C}^{n_1 \times \cdots \times n_d} \to \mathbb{C}^{n_1 \times \cdots \times n_d}, \quad \mathcal{H} = \sum_{k=1}^{d} \mathcal{D}^{(k)} + \sum_{1 \le i < j \le d} \beta(i,j) \mathcal{A}^{(i)} \mathcal{A}^{(j)}. \tag{3.1}$$

Here, $\mathcal{D}^{(i)}$ and $\mathcal{A}^{(i)}$ represent operators that act on the ith site. The matrix representation of a single-site operator like $\mathcal{A}^{(i)}$ takes the form $\mathbf{I}_{n_d} \otimes \cdots \otimes \mathbf{I}_{n_{i+1}} \otimes \mathbf{A}_i \otimes \mathbf{I}_{n_{i-1}} \otimes \cdots \otimes \mathbf{I}_{n_1}$ for some matrix $\mathbf{A}_i \in \mathbb{C}^{n_k \times n_k}$, where $\mathbf{I}$ denotes an identity matrix of suitable size and $\otimes$ denotes the usual Kronecker product. For quantum spin systems, the matrices $\mathbf{A}_i$ are usually the Pauli matrices. The coefficients $\beta(i,j)$ characterize the strength of the interaction between the ith and the jth particle. A sparse interaction matrix $\boldsymbol{\beta}$, containing the coefficients $\beta(i,j)$, encodes that only a few sites interact with each other, while a banded matrix $\boldsymbol{\beta}$ encodes short-range interactions. If $\boldsymbol{\beta}$ is not banded, also distant sites interact. This regime of long-range interactions significantly complicates the use of tensor network methods, including the compact representation of the underlying Hamiltonian.

Considering an example from many-body quantum physics, the main computational costly part often is the application of the corresponding Hamiltonian to the state as well as the memory footprint to store the Hamiltonian. For a Hamiltonian describing a quantum spin system of d two-level spin $\frac{1}{2}$ particles, the full Hamiltonian is a matrix of size $2^d \times 2^d$. Without using any structure or sparsity, the cost of saving the full matrix scales exponentially with the system size. I.e. for d particles, we have the following memory footprint in double precision for storing the full Hamiltonian:

Number of particles	$d = 8$	$d = 16$	$d = 32$
Memory requirement	524KB	34.3GB	147EB (= exabytes)

TABLE 3.1: Memory footprint of full Hamiltonian

Thus, it is of interest to find compact representations of the corresponding Hamiltonians. There is a broad literature on how to efficiently represent an operator in tensor train/matrix product state representation. These objects are usually referred to as matrix product operators (MPOs). We refer to [Sch11] and [PMCV10] for their construction and further details. It is well-known that ranks are constant for next-neighbor and/or translation-invariant interactions. In the broader context of partial differential equations, MPO representations for $\mathcal{H}$ have been derived in [DK13, KRS13], noting an interesting connection between the MPO representation and the quasi-separability of $\boldsymbol{\beta}$. We refer to [EG99] for more details on quasi-separability.

The construction of the more general class of tree tensor network operators for nearest neighbor interactions was discussed in [Tob12]. Recently, state diagrams have been used to construct quantum Hamiltonians in tree tensor network representation in [MHM24b]. In [LRY+24] the authors construct TTNOs and in [RLJS20] MPOs by finding the minimum vertex cover of a bipartite graph. However, in this chapter, we will extend the work of Lin and Tong [LT21] from MPOs to general TTNOs. The authors use hierarchical low-rank decompositions to find efficient constructions for MPOs based on the HODLR format and the $\mathcal{H}$-matrix format. In this work, we promote the usage of HSS matrices whose structure is fundamentally close to the one of TTNs.

Remark 3.1. We note that an operator as in (3.1) might also arise from the discretization of partial differential equations (PDEs) on d-dimensional hypercube. Thus, the applicability of such techniques is not restricted to quantum many-body problems only but can be incorporated in various applications

3.1 Formalism of Tree Tensor Network Operators

Recall definition 2.3 of a tree with unequal leaves and fix a tree $\bar{\tau}$ together with a subtree $\tau = (\tau_1, \dots, \tau_m)$. In the following, we will additionally assume that

$$L(\tau_i) < L(\tau_j) \quad \forall i < j. \tag{3.2}$$

By assumption (3.2), we know that for any subtree $\tau = (\tau_1, \dots, \tau_m) < \bar{\tau}$, there exist integers $l_1 < l_2 \cdots < l_{m+1}$ such that

$$L(\tau_1) = \{l_1, l_1 + 1, \dots, l_2 - 1\}, \quad \dots, \quad L(\tau_m) = \{l_m, l_m + 1, \dots, l_{m+1} - 1\},$$

i.e. the sets $L(\tau_i)$ consist of consecutive integers. We now introduce the definition of a tree tensor network operator as provided in [CKS24, Definition 2.4].

Definition 3.1: Tree Tensor Network Operator

Let $\mathcal{H} : \mathbb{C}^{n_1 \times \cdots \times n_d} \to \mathbb{C}^{n_1 \times \cdots \times n_d}$ be a linear operator, and $\widehat{H} \in \mathbb{C}^{(n_1 \dots n_d) \times (n_1 \dots n_d)}$ its associated linear map, i.e., $\operatorname{vec}(\mathcal{H}(X)) = \widehat{H} \operatorname{vec}(X)$ for all $X \in \mathbb{C}^{n_1 \times \cdots \times n_d}$. We define $\mathcal{H}$ as a tree tensor network operator (TTNO) of maximal rank r on a given tree $\bar{\tau}$ if the reshape $H \in \mathbb{C}^{n_1^2 \times \cdots \times n_d^2}$ of $\widehat{H}$ forms a tree tensor network of rank at most r on $\bar{\tau}$. The tree tensor network H is the TTNO representation of $\mathcal{H}$.

3.1.1 Example: Rank one TTNO

We wish to provide further intuition on how tree tensor network operators look like and give an idea of how the application of a TTNO to a TTN is defined.
For that we consider a tree of height one which corresponds to Tucker tensors. Suppose we have a linear operator of the form

$$\mathcal{H}(X) = X \bigtimes_{i=1}^{d} \mathbf{A}_i \quad \text{with} \quad \mathbf{A}_i \in \mathbb{C}^{n_i \times n_i}.$$

Applying the unfolding formula (2.4) we arrive at the matrix representation $\widehat{H}$ of the operator $\mathcal{H}$,

$$\operatorname{vec}(\mathcal{H}(X)) = \mathbf{Mat}_0\left(\mathcal{H}(X)\right)^{\top} = \underbrace{(\mathbf{A}_d \otimes \cdots \otimes \mathbf{A}_1)}_{=\widehat{H}} \operatorname{vec}(X).$$

We obtain then a Tucker representation by reshaping $\mathrm{vec}(\widehat{H})$ into a $n_1^2 \times \cdots \times n_d^2$ tensor, resulting in the outer product representation

$$H = \mathrm{vec}(\mathbf{A}_1) \circ \cdots \circ \mathrm{vec}(\mathbf{A}_d) \in \mathbb{C}^{n_1^2 \times \cdots \times n_d^2}, \tag{3.3}$$

see for example [KT14, Sch11]. Equation (3.3) corresponds to a rank-1 Tucker tensor $H = C \times_{i=1}^{d} \mathrm{vec}(\mathbf{A}_i)$ with $C = 1$. A summation of rank-1 operators allows for more complex linear operators. The idea of how to construct such rank-1 linear operators in TTNO format directly extends to general tree formats by setting the ith leaf to $\mathrm{vec}(\mathbf{A}_i)$ and all core tensors to 1.

Many linear operators can be written as a sum of rank one operators, i.e.

$$\mathcal{H}(X) = \sum_{k=1}^{s} X \bigtimes_{i=1}^{d} \mathbf{A}_k^{(i)}.$$

For example, the discretized Laplacian can be written in this shape. Let $\boldsymbol{D}$ be the discretization of the derivative. Then the discretized Laplacian can be written as

$$\boldsymbol{D}^{(d)} \otimes \mathbf{I} \otimes \cdots \otimes \mathbf{I} + \mathbf{I} \otimes \boldsymbol{D}^{(d-1)} \otimes \mathbf{I} \otimes \cdots \otimes \mathbf{I} + \mathbf{I} \otimes \cdots \otimes \mathbf{I} \otimes \boldsymbol{D}^{(1)}.$$

A straightforward way to construct the full TTNO is to compute all rank one TTNOs $X \times_{i=1}^{d} \mathbf{A}_k^{(i)}$ for $k = 1, \cdots, s$, by the strategy above. The resulting s TTNOs are then summed, recall subsection 2.4.1. However, this representation of the Hamiltonian is usually far from optimal in terms of the needed tree ranks and memory requirements. By this straightforward strategy, we would find a rank d TTNO representation of the discretized Laplacian, whereas it is known that it can be represented exactly by only using a tree rank of 2, cf. [Tob12, Example 3.7]. The example makes it clear that there is a lot of potential to find compact low-rank representations of such operators. Before we start the discussion about the compression of TTNOs, we wish to provide how the application of a TTNO to a TTN works.

3.1.2 Application of TTNOs

Suppose we want now to apply a tree tensor network operator to a tree tensor network. For that, we recall [KT14, § 8] how to efficiently apply a linear operator $\mathcal{H}$ in TTNO representation H to a tree tensor network X, both defined on the same tree $\bar{\tau}$. The evaluation of $\mathcal{H}(X)$ results again in a tree tensor network (of larger rank) and it can be efficiently computed as follows:

(i) For each leaf $\tau = l$, we set

$$(\mathcal{H}(X))_l = \left[\mathbf{A}_1^l \mathbf{U}_l, \dots, \mathbf{A}_s^l \mathbf{U}_l\right],$$

where $\mathbf{A}_j^l \in \mathbb{C}^{n_l \times n_l}$ is the matrix obtained from reshaping the jth column of the basis matrix at the lth leaf of the TTNO H.

(ii) For each subtree $\tau \in \mathcal{T}(\bar{\tau})$, we set

$$C_\tau^{\mathcal{H}(X)} = C_\tau^H \otimes C_\tau^X. \tag{3.4}$$

where C_τ^H denotes the core tensor of the TTNO representation H at τ, and C_τ^X denotes the core tensor of the TTN X at τ.

The operation (3.4) involves the Kronecker product of tensors. We recall that the tensor Kronecker product $C \in \mathbb{C}^{(n_1 m_1) \times \cdots \times (n_d m_d)}$ of tensors $A \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ and $B \in \mathbb{C}^{m_1 \times \cdots \times m_d}$ is defined element-wise for $1 \le i_k \le n_k$ and $1 \le j_k \le m_k$ via the relation

$$C(j_1 + (i_1 - 1)m_1, \dots, j_d + (i_d - 1)m_d) := A(i_1, \dots, i_d) B(j_1, \dots, j_d).$$

Note that the above operations (i) and (ii) are independent and can be performed fully in parallel. If H has tree ranks $(s_\tau)_{\tau \le \bar{\tau}}$ and X has tree ranks $(r_\tau)_{\tau \le \bar{\tau}}$, then the resulting tree tensor network $\mathcal{H}(X)$ has tree ranks $(s_\tau r_\tau)_{\tau \le \bar{\tau}}$. This means that operation (ii) multiplies each rank of the original tensor network by the corresponding rank of the TTNO. Hence, it is crucial to find a low-rank TTNO representation H.

3.2 Unstructured case

In this section we want to derive a construction of a TTNO for Hamiltonian systems as in (3.1), where no assumptions on the interaction matrix $\boldsymbol{\beta}$ are made. The operator consists of two sums - one Laplacian-like part which acts on individual sites and the interaction term. It is well-known how to construct a TTNO for the Laplacian-like part, see above or for example [KT14]. We briefly give the basis matrices and core tensors in this case:

$$\mathbf{U}_l = [\mathrm{vec}(\mathcal{D}^{(l)}), \mathrm{vec}(\mathbf{I})] \quad \forall l \in \mathcal{L},$$

$$C_\tau(i_1, \dots, i_m) = \begin{cases} 1 & \text{if } i_1 = \cdots = i_m = 1, \\ 0 & \text{else,} \end{cases}$$

where $C_\tau \in \mathbb{C}^{2\times\cdots\times 2}$. Note that the resulting TTNO is of rank two for every subtree $\tau \leq \bar{\tau}$. The construction of a TTNO for the interaction part of (3.1) is more complicated. Thus, in the following, we will investigate the interaction part, which has a matrix representation of the form

$$\widehat{H} = \sum_{1\leq i<j\leq d} \beta(i,j)\cdot \mathbf{A}^{(i)}\mathbf{A}^{(j)} \in \mathbb{C}^{(n_1\cdots n_d)\times(n_1\cdots n_d)}, \tag{3.5}$$

with $\mathbf{A}^{(k)} = \mathbf{I}_{n_d}\otimes\cdots\otimes\mathbf{I}_{n_{k+1}}\otimes\mathbf{A}_k\otimes\mathbf{I}_{n_{k-1}}\otimes\cdots\otimes\mathbf{I}_{n_1}$ and $\mathbf{A}_k \in \mathbb{C}^{n_k\times n_k}$ for $k = 1,\dots,d$. We recall that the interaction matrix $\boldsymbol{\beta} = (\beta(i,j))_{i<j}$ is an upper triangular matrix whose coefficients encode the strength of the interaction between the sites. For example, $\beta(i,j) = \frac{1}{|i-j|}$ encodes Coulomb interactions between particles while $\beta(i,j) = \frac{1}{|i-j|^3}$ encodes dipole-dipole interactions [SWM10].

3.2.1 Binary tree tensor network operators

To simplify the presentation, we first consider binary trees $\bar{\tau}$ for the construction. The general case is discussed in the next subsection.
To obtain a TTNO representation $H \in \mathbb{C}^{n_1^2\times\cdots\times n_d^2}$ we vectorize the summands in (3.5)

$$\boldsymbol{h} := \mathrm{vec}(\widehat{H}) = \sum_{1\leq i<j\leq d} \beta(i,j)\cdot \mathbf{a}^{(i,j)} \in \mathbb{C}^{n_1^2\cdots n_d^2}.$$

where

$$\mathbf{a}^{(i,j)} := \mathbf{e}_d\otimes\cdots\otimes\mathbf{e}_{j+1}\otimes\mathrm{vec}(\mathbf{A}_j)\otimes\mathbf{e}_{j-1}\otimes\cdots\otimes\mathbf{e}_{i+1}\otimes\mathrm{vec}(\mathbf{A}_i)\otimes\mathbf{e}_{i-1}\otimes\cdots\otimes\mathbf{e}_1, \tag{3.6}$$

and $\mathbf{e}_k = \mathrm{vec}(\mathbf{I}_{n_k})$ denotes the vectorization of the identity matrix. For the recursive structure of the TTNO is it convenient to define one- and two-site matrix representations. I.e. for a tree τ with $i,j \in L(\tau)$ we set

$$\mathbf{a}_\tau^{(i)} = \bigotimes_{\substack{l\in L(\tau)\\ l>i}} \mathbf{e}_l \otimes \mathrm{vec}(\mathbf{A}_i) \bigotimes_{\substack{l\in L(\tau)\\ l<i}} \mathbf{e}_l \in \mathbb{C}^{n_\tau^2}, \tag{3.7}$$

$$\mathbf{a}_\tau^{(i,j)} = \bigotimes_{\substack{l\in L(\tau)\\ l>j}} \mathbf{e}_l \otimes \mathrm{vec}(\mathbf{A}_j) \overset{i+1}{\bigotimes_{\substack{l\in L(\tau)\\ i<l<j}}} \mathbf{e}_l \otimes \mathrm{vec}(\mathbf{A}_i) \bigotimes_{\substack{l\in L(\tau)\\ l<i}} \mathbf{e}_l \in \mathbb{C}^{n_\tau^2}. \tag{3.8}$$

The Kronecker products are executed in decreasing order with respect to l. This will be used to represent the operator which only acts on the subsystem, where only the sites $i,j \in L(\tau)$ are interacting. Note that $\mathbf{a}_\tau^{(i,j)}$ matches $\mathbf{a}^{(i,j)}$ from (3.6) for $\tau = \bar{\tau}$.
Consider now a subtree $\tau \leq \bar{\tau}$. Using the two-site matrix representation (3.8), the

vectorized Hamiltonian for τ takes the form

$$\mathbf{h}_\tau := \sum_{\substack{i<j \\ i,j\in L(\tau)}} \beta(i,j)\cdot \mathbf{a}_\tau^{(i,j)} \in \mathbb{C}^{n_\tau^2}, \tag{3.9}$$

with $n_\tau = \prod_{i\in L(\tau)} n_i$. By definition (3.8) it is ensured that $\mathbf{A}_i$ and $\mathbf{A}_j$ still act on the ith and jth site, respectively. If we are at a leaf we set $\mathbf{h}_l = 0$, while at $\tau = \bar{\tau}$ we obtain $\mathbf{h}_\tau = \mathbf{h}$.

The following Lemma [CKS24, Lemma 3.1] allows us to construct the TTNO representation recursively. For readability, we introduce the short notation $\mathbf{e}_\tau := \bigotimes_{l\in L(\tau)} \mathbf{e}_l$.

Lemma 3.1

For $\tau = (\tau_1, \tau_2) \leq \bar{\tau}$, the vector $\mathbf{h}_\tau$ defined in (3.9) satisfies

$$\mathbf{h}_\tau = \mathbf{e}_{\tau_2}\otimes \mathbf{h}_{\tau_1} + \mathbf{h}_{\tau_2}\otimes \mathbf{e}_{\tau_1} + \sum_{\substack{i\in L(\tau_1) \\ j\in L(\tau_2)}} \beta(i,j)\cdot \mathbf{a}_{\tau_2}^{(j)}\otimes \mathbf{a}_{\tau_1}^{(i)}, \tag{3.10}$$

with $\mathbf{a}_{\tau_1}^{(i)}, \mathbf{a}_{\tau_2}^{(j)}$ defined as in (3.7).

Proof. The disjoint union $L(\tau) = L(\tau_1) \dot\cup L(\tau_2)$ induces the partition

$$L(\tau)\times L(\tau) = L(\tau_1)\times L(\tau_1) \cup L(\tau_2)\times L(\tau_2) \cup L(\tau_1)\times L(\tau_2) \cup L(\tau_2)\times L(\tau_1). \tag{3.11}$$

We now consider the corresponding division of the sum (3.9) defining $\mathbf{h}_\tau$.

First and second subset: The first subset of (3.11) only considers terms in the sum which come from the same subtree, i.e. for which $(i,j) \in L(\tau_1)\times L(\tau_1)$. By using that (3.7) implies $\mathbf{a}_\tau^{(i,j)} = \mathbf{e}_{\tau_2}\otimes \mathbf{a}_{\tau_1}^{(i,j)}$ for $i,j \in L(\tau_1)$, we obtain

$$\sum_{\substack{i<j \\ i,j\in L(\tau_1)}} \beta(i,j)\cdot \mathbf{a}_\tau^{(i,j)} = \mathbf{e}_{\tau_2}\otimes \sum_{\substack{i<j \\ i,j\in L(\tau_1)}} \beta(i,j)\cdot \mathbf{a}_{\tau_1}^{(i,j)} = \mathbf{e}_{\tau_2}\otimes \mathbf{h}_{\tau_1},$$

which equals the first term in (3.10). Analogously, we obtain that the second subset $L(\tau_2)\times L(\tau_2)$ in (3.11) yields the second term $\mathbf{h}_{\tau_2}\otimes \mathbf{e}_{\tau_1}$ in (3.10).

Third and fourth subset: The third subset in (3.11) corresponds directly to the third term in (3.10). The fourth subset does not contribute any terms as there exists no index pair $(i,j) \in L(\tau_2)\times L(\tau_1)$ such that the condition $i<j$ is satisfied. □

Before proving one of the main results from this chapter, we first prove a generalization of lemma 3.1. It will be a purely theoretical result, which is not needed for the actual construction of the TTNO, but will help prove the main theorem. We will split a tree

into a subtree τ and its complement $\bar{\tau} \setminus \tau$. This situation is more complicated compared to the previous lemma, as the leaves of τ are not necessarily smaller than the leaves of $\bar{\tau}$. To formulate the generalization it is convenient to introduce the symmetrized interaction matrix

$$\boldsymbol{\beta}_s := \boldsymbol{\beta} + \boldsymbol{\beta}^\mathsf{T}. \tag{3.12}$$

The coefficients of $\boldsymbol{\beta}_s$ then fulfill

$$\begin{aligned}
\boldsymbol{\beta}_s(i,j) &= \boldsymbol{\beta}(i,j) \text{ for } i < j, \\
\boldsymbol{\beta}_s(i,j) &= \boldsymbol{\beta}(j,i) \text{ for } i > j, \\
\boldsymbol{\beta}_s(i,i) &= 0 \ \forall i.
\end{aligned}$$

With this we can formulate now the generalized lemma. The statement and its proof can be found as lemma 3.2 in [CKS24].

Lemma 3.2

For $\tau \in \mathcal{T}(\bar{\tau}) \cup L(\bar{\tau})$, consider a permutation of the modes that puts the leaves of τ first:

$$\underbrace{l_1, l_1+1, \ldots, l_2-1}_{L(\tau)}, \underbrace{1, 2, \ldots, l_1-1, l_2, l_2+1, \ldots, d}_{L(\bar{\tau}\setminus\tau)}.$$

Let $\widehat{H}_{\tau,\bar{\tau}\setminus\tau}$ denote the corresponding permutation of the matrix representation $\widehat{H}$ from (3.5). Then the vectorization of $\widehat{H}_{\tau,\bar{\tau}\setminus\tau}$ takes the form

$$\mathbf{h}_{\tau,\bar{\tau}\setminus\tau} = \mathbf{e}_{\bar{\tau}\setminus\tau} \otimes \mathbf{h}_\tau + \mathbf{h}_{\bar{\tau}\setminus\tau} \otimes \mathbf{e}_\tau + \sum_{\substack{i\in L(\tau) \\ j\in L(\bar{\tau}\setminus\tau)}} \beta_s(i,j) \cdot \mathbf{a}^{(j)}_{\bar{\tau}\setminus\tau} \otimes \mathbf{a}^{(i)}_\tau \in \mathbb{C}^{n_\tau^2 \cdot n^2_{\bar{\tau}\setminus\tau}}, \tag{3.13}$$

where $\mathbf{h}_{\bar{\tau}\setminus\tau} \in \mathbb{C}^{n^2_{\bar{\tau}\setminus\tau}}$ is the vectorization of the Hamiltonian that only considers interactions between sites contained in $L(\bar{\tau} \setminus \tau)$ and $n_{\bar{\tau}\setminus\tau} = \prod_{j\in L(\bar{\tau}\setminus\tau)} n_j^2$. The vectors $\mathbf{a}^{(i)}_\tau$, $\mathbf{a}^{(j)}_{\bar{\tau}\setminus\tau}$ are defined as in (3.7).

Proof.

We will prove the statement with similar arguments as in the proof of Lemma 3.1. Analogously to (3.11), the partition is induced by the disjoint union $L(\tau) = L(\tau) \dot\cup L(\bar{\tau} \setminus \tau)$:

$$\{1,\ldots,d\} \times \{1,\ldots,d\} = L(\tau)\times L(\tau) \cup L(\bar{\tau}\setminus\tau)\times L(\bar{\tau}\setminus\tau) \cup L(\tau)\times L(\bar{\tau}\setminus\tau) \cup L(\bar{\tau}\setminus\tau)\times L(\tau).$$

With the same explanation as in the proof of Lemma 3.1, the first two terms of this partition match the first two terms of (3.13). As $\beta(i,j) = 0$ for $i \geq j$ and using the

definition of the symmetrized interaction matrix $\boldsymbol{\beta}_s$, the last two terms give

$$\begin{aligned}
&\sum_{i\in L(\tau),j\in L(\bar{\tau}\setminus\tau),i<j} \beta(i,j)\cdot \mathbf{a}^{(j)}_{\bar{\tau}\setminus\tau}\otimes\mathbf{a}^{(i)}_{\tau} + \sum_{i\in L(\bar{\tau}\setminus\tau),j\in L(\tau),i<j} \beta(i,j)\cdot \mathbf{a}^{(i)}_{\bar{\tau}\setminus\tau}\otimes\mathbf{a}^{(j)}_{\tau}\\
=&\sum_{i\in L(\tau),j\in L(\bar{\tau}\setminus\tau),i<j} \beta(i,j)\cdot \mathbf{a}^{(j)}_{\bar{\tau}\setminus\tau}\otimes\mathbf{a}^{(i)}_{\tau} + \sum_{i\in L(\tau),j\in L(\bar{\tau}\setminus\tau),i>j} \beta(j,i)\cdot \mathbf{a}^{(j)}_{\bar{\tau}\setminus\tau}\otimes\mathbf{a}^{(i)}_{\tau}\\
=&\sum_{i\in L(\tau),j\in L(\bar{\tau}\setminus\tau)} \beta_s(i,j)\cdot \mathbf{a}^{(j)}_{\bar{\tau}\setminus\tau}\otimes\mathbf{a}^{(i)}_{\tau}.
\end{aligned}$$

□

Finally, we are now able to prove a first bound on the ranks that are needed to represent a Hamiltonian in a TTNO format. The following theorem and its proof can be found in [CKS24, Theorem 3.3].

Theorem 3.1: TTNO unstructred

Let $\widehat{H}$ be the linear operator defined by (3.5) and let $\bar{\tau}$ be a binary tree. Then $\widehat{H}$ admits a TTNO representation H such that the ranks r_τ satisfy $r_l = 2$ at every leaf $l \in L(\bar{\tau})$ and

$$r_\tau = 2 + \operatorname{rank}(\boldsymbol{\beta}_s(\tau, \bar{\tau}\setminus\tau)) \le 2 + d_\tau, \quad \forall \tau \in T(\bar{\tau})\setminus\bar{\tau}, \tag{3.14}$$

where d_τ denotes the cardinality of $L(\tau)$ and $\boldsymbol{\beta}_s(\tau, \bar{\tau}\setminus\tau)$ is the $d_\tau \times (d - d_\tau)$ submatrix obtained by selecting the rows in $L(\tau)$ and the columns in $\{1,\dots,d\}\setminus L(\tau)$ of the symmetrized interaction matrix $\boldsymbol{\beta}_s$ from (3.12).

Proof. By definition 3.1, theorem 2.3 and the notation introduced there for $\mathbf{Mat}_{L(\tau)}$, we know that for each $\tau \le \bar{\tau}$, $H \in \mathbb{C}^{n_1^2\times\cdots\times n_d^2}$ admits a TTNO representation with the rank r_τ given by the rank of the matricization $\mathbf{Mat}_{L(\tau)}(H)$. By lemma 3.2 we have that

$$\mathbf{Mat}_{L(\tau)}(H) = \mathbf{h}_\tau\ \mathbf{e}^{\top}_{\bar{\tau}\setminus\tau} + \mathbf{e}_\tau\ \mathbf{h}^{\top}_{\bar{\tau}\setminus\tau} + \sum_{\substack{i\in L(\tau)\\ j\in L(\bar{\tau}\setminus\tau)}} \beta_s(i,j)\cdot \mathbf{a}^{(i)}_{\tau}\ \left(\mathbf{a}^{(j)}_{\bar{\tau}\setminus\tau}\right)^{\top}. \tag{3.15}$$

For a leaf $\tau = l$, this simplifies to

$$\mathbf{a}_l\ \mathbf{e}^{\top}_{\bar{\tau}\setminus l} + \mathbf{e}_l\ \mathbf{h}^{\top}_{\bar{\tau}\setminus\tau} + \sum_{j\neq l} \beta_s(l,j)\cdot \mathbf{a}_l\ \left(\mathbf{a}^{(j)}_{\bar{\tau}\setminus l}\right)^{\top},$$

which is of at most rank 2, i.e. $r_l = 2$. Now consider a general subtree $\tau \le \bar{\tau}$. We rewrite the third term in (3.15) as a matrix product of three matrices, the matrix containing the columns $\mathbf{a}^{(i)}_{\tau}$, the matrix $\boldsymbol{\beta}_s(\tau, \bar{\tau}\setminus\tau)$ and the matrix containing the rows $\left(\mathbf{a}^{(j)}_{\bar{\tau}\setminus\tau}\right)^{\top}$.

The rank of this matrix product is then bounded by the rank of $\boldsymbol{\beta}_s(\tau, \bar{\tau} \setminus \tau)$, which establishes (3.14). □

By theorem 3.1 we proved that we can construct a TTNO representation with $\lfloor d/2 \rfloor + 2$ as an upper bound for the tree ranks, as $d_\tau \leq \lfloor d/2 \rfloor$ for a balanced binary tree. It is noteworthy that even without any assumptions on the interaction matrix $\boldsymbol{\beta}$, we have only a linear growth of the ranks compared to a quadratic growth attained by the naive construction explained in subsection 3.1.1. The theory has been done for binary trees, hence the class of matrix product state/tensor trains is also covered, cf. chapter 2.

3.2.1.1 Construction of binary TTNOs

By theorem 3.1 we know that there exists a TTNO representation of the operator (3.1) of maximal rank $\lfloor d/2 \rfloor + 2$. In the following, we provide an explicit construction of the TTNO, which can be found in [CKS24, Section 3]. We start with a tree $\tau = (\tau_1, \tau_2)$, where neithter τ_1 nor τ_2 is a leaf. By enumerating the set of leaves $L(\tau_1) = \{l_1, \dots, l_2 - 1\}$ and $L(\tau_2) = \{l_2, \dots, l_3 - 1\}$, we introduce the matrices

$$\mathbf{U}_{\tau_1} := \begin{bmatrix} \mathbf{e}_{\tau_1} & \mathbf{h}_{\tau_1} & \mathbf{a}_{\tau_1}^{(l_1)} & \cdots & \mathbf{a}_{\tau_1}^{(l_2-1)} \end{bmatrix} \in \mathbb{C}^{n_{\tau_1}^2 \times (2+d_{\tau_1})},$$
$$\mathbf{U}_{\tau_2} := \begin{bmatrix} \mathbf{e}_{\tau_2} & \mathbf{h}_{\tau_2} & \mathbf{a}_{\tau_2}^{(l_2)} & \cdots & \mathbf{a}_{\tau_2}^{(l_3-1)} \end{bmatrix} \in \mathbb{C}^{n_{\tau_2}^2 \times (2+d_{\tau_2})}.$$

By formula (3.10) in lemma 3.1 we know that each column in the corresponding matrix $\mathbf{U}_\tau$ can be represented as a linear combination of Kronecker product between columns in $\mathbf{U}_{\tau_2}$ and $\mathbf{U}_{\tau_1}$. This implies the existence of a core tensor $C_\tau \in \mathbb{C}^{(2+d_{\tau_1}) \times (2+d_{\tau_2}) \times (2+d_\tau)}$ such that the matrix $\mathbf{U}_\tau$ has the form

$$\mathbf{U}_\tau = \begin{bmatrix} \mathbf{e}_\tau & \mathbf{h}_\tau & \mathbf{a}_\tau^{(l_1)} & \cdots & \mathbf{a}_\tau^{(l_3-1)} \end{bmatrix} = (\mathbf{U}_{\tau_2} \otimes \mathbf{U}_{\tau_1})\, \mathbf{Mat}_0(C_\tau)^\top \in \mathbb{C}^{n_\tau^2 \times (2+d_\tau)},$$

where $n_\tau = n_{\tau_1} n_{\tau_2}$. We determine the entries of the core tensor C_τ by giving all frontal slices $C(:,:,k)$ of C_τ, for $k = 1, \dots, (2 + d_\tau)$. To do so it is helpful to reshape each column of $\mathbf{U}_\tau$ as a $n_{\tau_1} \times n_{\tau_2}$ matrix. With $\mathbf{U}_\tau^{(k)}$, we denote the matrix corresponding to the kth column. By basic properties of the Kronecker product we have

$$\mathbf{U}_\tau^{(k)} = \mathbf{U}_{\tau_1} C_\tau(:,:,k) \mathbf{U}_{\tau_2}^\top.$$

With that we can start giving the slices of C_τ. As $\mathbf{e}_\tau$, the first column of $\mathbf{U}_\tau$, corresponds to the matrix $\mathbf{U}_\tau^{(1)} = \mathbf{e}_{\tau_1} \mathbf{e}_{\tau_2}^\top$, we obtain that the first slice $C_\tau(:,:,1)$ contains zero

everywhere except for the entry 1 at position $(1,1)$. I.e. we have

$$C_\tau(:,:,1) = \left[\begin{array}{cc|c} 1 & 0 & \\ 0 & 0 & \mathbf{0} \\ \hline & \mathbf{0} & \mathbf{0} \end{array}\right], \tag{3.16}$$

where the bold zeros are vectors/matrices only containing zeros of matching size. The second column of $\mathbf{U}_\tau$ encodes the interaction between sites in $L(\tau)$. The interactions between sites which are only contained in $L(\tau_1)$ are already encoded in the second column of $\mathbf{U}_{\tau_1}$. Hence, this column must be only multiplied with an identity from $\mathbf{U}_{\tau_2}$. Therefore we have one at the $(2,1)$ position of $C(:,:,2)$. Analogously, we get one at the $(1,2)$ position for the interactions between sites which are only contained in $L(\tau_2)$. The interactions among sites $\mathbf{a}_{\tau_1}^{(i)}$ in $L(\tau_1)$ and sites $\mathbf{a}_{\tau_2}^{(j)}$ in $L(\tau_2)$ are encoded in the matrix $\boldsymbol{\beta}(\tau_1,\tau_2)$. To describe these interactions correctly we set the lower-left block of $C_\tau(:,:,2)$ as $\boldsymbol{\beta}(\tau_1,\tau_2)$. In total, we obtain

$$C_\tau(:,:,2) = \left[\begin{array}{cc|c} 0 & 1 & \\ 1 & 0 & \mathbf{0} \\ \hline & \mathbf{0} & \boldsymbol{\beta}(\tau_1,\tau_2) \end{array}\right]. \tag{3.17}$$

With the first two slices we have constructed the identity and the interaction among all sites in $L(\tau)$ respectively. It remains to construct the single sites, which is needed to enable interactions between sites on higher levels of the TTNO. Thus, all single sites must be connected to an identity. We obtain the remaining $d_\tau = d_{\tau_1} + d_{\tau_2}$ slices of C_τ by

$$\left[\begin{array}{cc|c} 0 & 0 & \\ 0 & 0 & \mathbf{0} \\ \hline & \mathbf{0} & \mathbf{u}_3\mathbf{u}_1^\top \end{array}\right], \ldots, \left[\begin{array}{cc|c} 0 & 0 & \\ 0 & 0 & \mathbf{0} \\ \hline & \mathbf{0} & \mathbf{u}_{2+d_{\tau_1}}\mathbf{u}_1^\top \end{array}\right], \left[\begin{array}{cc|c} 0 & 0 & \\ 0 & 0 & \mathbf{0} \\ \hline & \mathbf{0} & \mathbf{u}_1\mathbf{u}_3^\top \end{array}\right], \ldots, \left[\begin{array}{cc|c} 0 & 0 & \\ 0 & 0 & \mathbf{0} \\ \hline & \mathbf{0} & \mathbf{u}_1\mathbf{u}_{2+d_{\tau_2}}^\top \end{array}\right], \tag{3.18}$$

where $\mathbf{u}_i$ is the vector (of appropriate length) with one at entry i and zeros everywhere else. For a more detailed explanation, consider the first matrix of (3.18). This matrix encodes the interaction between the l_1 site in τ_1 and the identity from τ_2. Therefore the slice produces the third column of $\mathbf{U}_\tau$, i.e. $\mathbf{a}_\tau^{(l_1)}$. The other slices are constructed following the same strategy.

We have constructed the full core tensor C_τ. It remains to construct the leaves $\mathbf{U}_l$ of the TTNO, i.e. when τ_1 and/or τ_2 is a leaf. Clearly, the interaction terms $\mathbf{h}_{\tau_i}$ from equation (3.10) vanish. Consequently, the basis matrices do not need to take these terms into account and only the identity and the action on the lth side $\mathbf{A}_l$ have to be

considered. The basis matrices then take the form

$$\mathbf{U}_l := \begin{bmatrix} \mathbf{e}_l & \operatorname{vec}(\mathbf{A}_l) \end{bmatrix} \in \mathbb{C}^{n_l^2 \times 2}.$$

Note that the core tensors connected to a leaf have to be adjusted to the structure of the basis matrices. It suffices to take into account the first slice of C_τ together with a modified version of the second slice, where one or both of the ones in the upper-right block are set to zero. At the root it is suffices to form $h_{\bar{\tau}}$, which is equal to $\mathbf{h}$ by construction. Hence, the core tensor at the root only consists of the matrix which equals (3.17).

The presented construction satisfies $r_l = 2$ for all leaves $l = 1, \dots, d$ and $r_\tau = d_\tau + 2$ for all $\tau \neq \bar{\tau}$. Hence, we found a construction of a TTNO which satisfies the bounds from theorem 3.1. Thus, we also proved that for Hamiltonians of the form (3.1) there always exists a matrix product operator (MPO) representation of maximal rank at most $\mathcal{O}(d)$.

3.2.2 Construction for general trees

We conclude this section by extending the previous results to general trees, cf. definition 2.3. Note that the explicit construction was not given in [CKS24] but is an additional contribution of this thesis.

The construction of the corresponding TTNO follows the same idea as for the binary case. Whilst the basis matrices do not change, the construction of the core tensors becomes slightly more technical. Suppose a tree $\tau = (\tau_1, \dots, \tau_m) \leq \bar{\tau}$. Analogously to the binary case, we partition the interaction matrix $\boldsymbol{\beta}$ into $\frac{m^2+m}{2}$ blocks

$$\boldsymbol{\beta}(\tau, \tau) = \begin{bmatrix} \boldsymbol{\beta}(\tau_1, \tau_1) & \cdots & \boldsymbol{\beta}(\tau_1, \tau_m) \\ & \ddots & \vdots \\ & & \boldsymbol{\beta}(\tau_m, \tau_m) \end{bmatrix}.$$

Similar to (3.10), this implies a partition for the vectorized Hamiltonian $\mathbf{h}_\tau$ of the form

$$\mathbf{h}_\tau = \sum_{i=1}^{m} \mathbf{e}_{\tau_m} \otimes \cdots \otimes \mathbf{e}_{\tau_{i+1}} \otimes \mathbf{h}_{\tau_i} \otimes \mathbf{e}_{\tau_{i-1}} \otimes \cdots \otimes \mathbf{e}_{\tau_1} + \sum_{i<j}^{m} \sum_{\substack{k \in L(\tau_i) \\ l \in L(\tau_j)}} \beta(k,l) \cdot \mathbf{a}_{\tau_j}^{(l)} \otimes \mathbf{a}_{\tau_i}^{(k)}.$$

We enumerate the sites by $L(\tau_i) = \{l_i, l_i + 1, \dots, l_{i+1} - 1\}$ and define as above

$$\mathbf{U}_{\tau_i} = \begin{bmatrix} \mathbf{e}_{\tau_i} & \mathbf{h}_{\tau_i} & \mathbf{a}^{(l_i)} & \cdots & \mathbf{a}^{(l_{i+1}-1)} \end{bmatrix}, \qquad i = 1, \dots, m.$$

In analogy to the binary tree, there exists a core tensor C_τ of order $m+1$ such that

$$\mathbf{U}_\tau = \begin{bmatrix} \mathbf{e}_\tau & \mathbf{h}_\tau & \mathbf{a}^{(l_1)} & \cdots & \mathbf{a}^{(l_{m+1}-1)} \end{bmatrix} = (\mathbf{U}_{\tau_m} \otimes \cdots \otimes \mathbf{U}_{\tau_1})\, \mathbf{Mat}_0(C_\tau)^\top.$$

It remains to determine the core tensor C_τ. As C_τ is an order $m+1$ tensor, we do not use slices of C_τ anymore. Instead, we determine the columns of its matricization $\mathbf{Mat}_0(C_\tau)^\top$.

Identity

The first column again encodes the identity and hence is just the first unit vector of matching size, i.e.

$$\mathbf{Mat}_0(C_\tau)^\top(:,1) = \mathbf{e}_1 \in \mathbb{C}^{(\prod_{i=1}^{m} d_{\tau_i}+2)+2}.$$

Interactions

The second column of $\mathbf{Mat}_0(C_\tau)^\top$ again only contains the interactions among sites. We start by only considering interactions of sites which lie in the same subtree. Note that this kind of interaction only exists if τ_i is not a leaf. Thus, for all τ_i which are not a leaf, we set the jth entry of the second columns of $\mathbf{Mat}_0(C_\tau)^\top$ to one if

$$\mathbf{Mat}_0(C_\tau)^\top(j,2) = 1,$$
$$\text{for } j = \prod_{k=1}^{i}(l_{k+1}-l_k) - \prod_{k=1}^{i-1}(l_{k+1}-l_k) + 1 \quad \text{with } i \in \{1,\dots,m\},$$

where the empty product is defined as one. With this, we have encoded all interactions coming from the same subtree. Now we need to take care of interactions of sites between different subtrees.

Suppose two sites i_k in the subtree τ_i and j_k in the subtree τ_j, with $i < j$. Further, let i_k be the s_ith leaf in the subtree τ_i and j_k be the s_jth leaf in the subtree τ_j respectively. Finally, define

$$p_1 := \left(\prod_{k=1}^{i}(l_{k+1}-l_k)\right)(s_i-1)+1$$
$$p_2 := \left(\prod_{k=i+1}^{j-1}(l_{k+1}-l_k)\right)(s_j-1)+1.$$

We want to describe now the interaction between the two sites i_k and j_k. This interaction is encoded by setting

$$\mathbf{Mat}_0(C_\tau)^\top(j,2) = \beta(i_k, j_k) \quad \text{for } j = \left(\prod_{k=1}^{i}(l_{k+1} - l_k)\right)(p_2 - 1) + p_1.$$

This is done for all possible combinations of i_k and j_k which fulfill the assumptions from above.

Single site actions

It remains to construct all single sites $i_k \in \{l_1, \dots, l_m + 1\}$ for encoding interactions on higher levels of the TTNO. Assume the i_kth leaf lies in the subtree τ_i and within τ_i it is the s_ith leaf. To encode the single action we set

$$\mathbf{Mat}_0(C_\tau)^\top(j, i_k + 2) = 1 \quad \text{for } j = \left(\prod_{k=1}^{i-1}(l_{k+1} - l_k)\right)(s_i - 1) + 1.$$

As for the binary trees, at the root, the core tensor consists only of the second column of $\mathbf{Mat}_0(C_\tau)^\top$.

3.3 Construction via Hierarchical Semi-Separable matrices

In the previous section a tree tensor network operator representation was found, where the ranks only scale linearly with the number of particles/sites. Along the construction, we did not use any property or structure of the interaction matrix $\boldsymbol{\beta}$. In this section, we will show that under certain assumptions on $\boldsymbol{\beta}$, it is possible to further reduce the ranks of the TTNO. We achieve this by decomposing the interaction matrix by a hierarchical semi-separable decomposition. This section generalizes the work of Lin and Tong on matrix product operators (MPOs), who also made use of hierarchical matrices [LT21].

3.3.1 Recap: Hierarchical semi-separable matrices

The Hierarchical Semi-Separabel (HSS) decomposition for matrices, as defined in [XCGL10], is an effective way to compactly store matrices with a hierarchical low-rank structure.

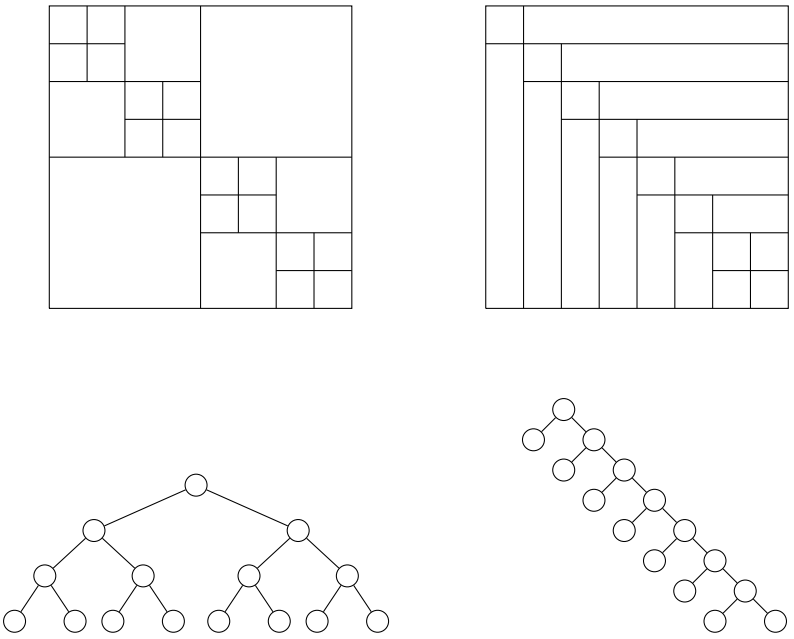

FIGURE 3.1: Different recursive block-partitions of an 8×8 interaction matrix $\boldsymbol{\beta}$. Left: Recursive block-partition corresponding to a balanced binary tree. Right: Recursive block-partition corresponding to a unbalanced binary tree.

Effective algorithms for large-scale problems have been derived for problems from numerical linear algebra, see [MRK20] and the references therein. Although the $d \times d$ interaction matrix $\boldsymbol{\beta}$ is of relatively small size, we will see in section 3.4 that many matrices $\boldsymbol{\beta}$ stemming from quantum spin systems admit a hierarchical low-rank structure. Exploiting this fact allows us to construct TTNOs with further reduced ranks.

The hierarchical structure of the decomposition is encoded through a binary tree $\bar{\tau}$ (cf. definition 2.3) and used to recursively block-partition $\boldsymbol{\beta}$. At the root $\bar{\tau} = (\tau_1, \tau_2)$, this corresponds to the partitioning

$$\boldsymbol{\beta} = \boldsymbol{\beta}(\bar{\tau}, \bar{\tau}) = \begin{bmatrix} \boldsymbol{\beta}(\tau_1, \tau_1) & \boldsymbol{\beta}(\tau_1, \tau_2) \\ & \boldsymbol{\beta}(\tau_2, \tau_2) \end{bmatrix}.$$

The partitioning is recursively repeated for the blocks $\boldsymbol{\beta}(\tau_1, \tau_1)$ and $\boldsymbol{\beta}(\tau_2, \tau_2)$ using the subtrees τ_1 and τ_2 respectively. The size of each of the blocks can be chosen related to the problem. In figure 3.1 a graphical representation of different tree structures and their related block partition is provided. The figure is taken from [CKS24, Figure 2]. The first tree on the left corresponds to the hierarchical Tucker format while the degenerated tree on the right corresponds to a tensor train/matrix product state. The latter is closely related to the notion of quasi-separable matrices, see [KMR19] for a discussion on this.

While the diagonal blocks on each level of the tree are further decomposed, each off-diagonal block $\boldsymbol{\beta}(\tau_1, \tau_2)$ is assumed to admit a low-rank factorization

$$\boldsymbol{\beta}(\tau_1, \tau_2) = \mathbf{V}_{\tau_1} \mathbf{S}_{\tau_1,\tau_2} \mathbf{V}_{\tau_2}^*, \qquad \mathbf{V}_{\tau_1} \in \mathbb{C}^{d_{\tau_1} \times k_{\tau_1}}, \ \mathbf{S}_{\tau_1,\tau_2} \in \mathbb{C}^{k_{\tau_1} \times k_{\tau_2}}, \ \mathbf{V}_{\tau_2} \in \mathbb{C}^{d_{\tau_2} \times k_{\tau_1}}, \quad (3.19)$$

for small integers k_{τ_1}, k_{τ_2}. The crucial assumption for HSS matrices is that the matrices $\mathbf{V}_{\tau_i}$ are nested across all levels. For each $\tau = (\tau_1, \tau_1)$, the nestedness is characterized by the existence of a translation operator $\boldsymbol{R}_\tau \in \mathbb{C}^{(k_{\tau_1}+k_{\tau_2}) \times k_\tau}$ such that

$$\mathbf{V}_\tau = \begin{pmatrix} \mathbf{V}_{\tau_1} & 0 \\ 0 & \mathbf{V}_{\tau_2} \end{pmatrix} \boldsymbol{R}_\tau \in \mathbb{C}^{d_\tau \times k_\tau}. \quad (3.20)$$

Recall that d_τ denotes the cardinality of $L(\tau)$. The maximal value of k_τ across all levels is defined as the so-called *HSS rank*. We will see later that the HSS rank, or generally all the k_τ on each level, encode the complexity of the corresponding TTNO. Thus, having only a hierarchical low-rank structure is enough to find memory-efficient representations of a Hamiltonian.

To fully reconstruct $\boldsymbol{\beta}$ from the factors of the HSS decomposition, it suffices to store only the middle factors $\mathbf{S}_{\tau_1,\tau_2}$ from (3.19) and the translation operators $\boldsymbol{R}_\tau$, if we assume normalized basis matrices on the leaves, i.e. $\mathbf{V}_l = 1$ for all $l \in \mathcal{L}$.

Remark 3.2. The left and right factors $\mathbf{V}_{\tau_i}$ from (3.19) are enforced to be identical, which is not the case for the general notation for HSS matrices, cf. [XCGL10]. However, the definition of a HSS matrix from (3.19) coincides with the usual HSS definition applied to the symmetrized interaction matrix $\boldsymbol{\beta}_s$. The results from [XCGL10] imply that the smallest rank k_τ , for which an HSS decomposition of $\boldsymbol{\beta}$ is admittable, is given by

$$k_\tau = \operatorname{rank}\big(\boldsymbol{\beta}_s(\tau, \bar{\tau} \setminus \tau)\big), \quad \forall \tau \in T(\bar{\tau}), \quad (3.21)$$

where $\boldsymbol{\beta}_s(\tau, \bar{\tau} \setminus \tau)$ denots an HSS block row of $\boldsymbol{\beta}_s$.

3.3.2 Construction of TTNO via HSS matrices

We use the HSS decomposition of $\boldsymbol{\beta}$ to construct the tree tensor network operator. We are now able to proof one of the main results from this chapter, where we can find a sharper bound for the maximal rank of a TTNO. The following theorem and its proof can be found in [CKS24, Corollary 4.2].

Theorem 3.2

Let $\widehat{H}$ be the linear operator defined by (3.5), and let $\bar{\tau}$ be a binary dimension tree. If $\boldsymbol{\beta}$ admits an HSS decomposition (3.19)–(3.20) then there exists a TTNO representation H of $\widehat{H}$ with tree ranks $r_\tau = 2 + k_\tau$ for every subtree $\tau \in T(\bar{\tau})$ and $r_l = 2$ for every leaf $l \in L(\bar{\tau})$.

Proof. By theorem 3.1 we know that the TTNO can be constructed with tree ranks $r_\tau = 2 + \operatorname{rank}\big(\boldsymbol{\beta}_s(\tau, \bar{\tau} \setminus \tau)\big)$. Hence, by remark 3.2 we obtain that $r_\tau = 2 + k_\tau$, which concludes the proof. □

Theorem 3.2 only states the existence of a TTNO with tree ranks $k_\tau+2$ for each subtree τ. In the remainder of this subsection, we will explicitly construct the TTNO corresponding to an operator of the form (3.5) established by theorem 3.2. The construction was originally done in [CKS24, Section 4].

For $\tau = (\tau_1, \tau_2) \neq \bar{\tau}$, where neither τ_1 nor τ_2 is a leaf, we recall that by lemma 3.1, the vectorized Hamiltonian takes the form

$$\mathbf{h}_\tau = \mathbf{e}_{\tau_2} \otimes \mathbf{h}_{\tau_1} + \mathbf{h}_{\tau_2} \otimes \mathbf{e}_{\tau_1} + \sum_{\substack{i \in L(\tau_1) \\ j \in L(\tau_2)}} \beta(i,j) \cdot \mathbf{a}_{\tau_2}^{(j)} \otimes \mathbf{a}_{\tau_1}^{(i)}.$$

We introduce the short-hand notation

$$\mathbf{a}_{\tau_1} := [\mathbf{a}_{\tau_1}^{(l_1)}, \cdots, \mathbf{a}_{\tau_1}^{(l_2-1)}] \in \mathbb{C}^{n_{\tau_1}^2 \times d_{\tau_1}}, \quad \mathbf{a}_{\tau_2} := [\mathbf{a}_{\tau_2}^{(l_2)}, \cdots, \mathbf{a}_{\tau_2}^{(l_3-1)}] \in \mathbb{C}^{n_{\tau_2}^2 \times d_{\tau_2}}.$$

Using this short notation together with a basic property of the Kronecker product for matrices, we can rewrite the last summand of $\mathbf{h}_\tau$ as

$$\sum_{\substack{i \in L(\tau_1) \\ j \in L(\tau_2)}} \beta(i,j) \cdot \mathbf{a}_{\tau_2}^{(j)} \otimes \mathbf{a}_{\tau_1}^{(i)} = (\mathbf{a}_{\tau_2} \otimes \mathbf{a}_{\tau_1}) \operatorname{vec}(\boldsymbol{\beta}(\tau_1, \tau_2)). \tag{3.22}$$

In the HSS setting, the off-diagonal block is decomposed by $\boldsymbol{\beta}(\tau_1, \tau_2) = \mathbf{V}_{\tau_1} \mathbf{S}_{\tau_1,\tau_2} \mathbf{V}_{\tau_2}^*$, see (3.19). I.e. the HSS structure of the interaction matrix $\boldsymbol{\beta}$ induces a low-rank structure of $\boldsymbol{\beta}(\tau_1, \tau_2)$. Inserting the decomposition in (3.22) we obtain

$$(\mathbf{a}_{\tau_2} \otimes \mathbf{a}_{\tau_1}) \operatorname{vec}(\boldsymbol{\beta}(\tau_1, \tau_2)) = (\mathbf{a}_{\tau_2} \mathbf{V}_{\tau_2} \otimes \mathbf{a}_{\tau_1} \mathbf{V}_{\tau_1}) \operatorname{vec}(\mathbf{S}_{\tau_1,\tau_2}). \tag{3.23}$$

Therefore, it suffices to consider the compressed bases

$$\tilde{\mathbf{a}}_{\tau_1} := \mathbf{a}_{\tau_1} \mathbf{V}_{\tau_1} \in \mathbb{C}^{n_{\tau_1}^2 \times k_{\tau_1}}, \quad \tilde{\mathbf{a}}_{\tau_2} := \mathbf{a}_{\tau_2} \mathbf{V}_{\tau_2} \in \mathbb{C}^{n_{\tau_2}^2 \times k_{\tau_2}}.$$

The remaining construction proceeds as for the unstructured case from section 3.2 but with the compressed bases. We introduce

$$\widetilde{\mathbf{U}}_{\tau_1} := [\mathbf{e}_{\tau_1},\ \mathbf{h}_{\tau_1},\ \tilde{\mathbf{a}}_{\tau_1}] \in \mathbb{C}^{n_{\tau_1}^2 \times (2+k_{\tau_1})}, \quad \widetilde{\mathbf{U}}_{\tau_2} := [\mathbf{e}_{\tau_2},\ \mathbf{h}_{\tau_2},\ \tilde{\mathbf{a}}_{\tau_2}] \in \mathbb{C}^{n_{\tau_2}^2 \times (2+k_{\tau_2})}.$$

Analogously to the unstructured case, we aim now to construct the corresponding core tensor $\widetilde{C}_\tau \in \mathbb{C}^{(2+k_{\tau_1})\times(2+k_{\tau_2})\times(2+k_\tau)}$ which transfers these bases to the corresponding basis at the parent node. There we want

$$\widetilde{\mathbf{U}}_\tau := [\mathbf{e}_\tau \quad \mathbf{h}_\tau \quad \tilde{\mathbf{a}}_\tau] \in \mathbb{C}^{n_\tau^2 \times (2+k_\tau)}, \qquad \tilde{\mathbf{a}}_\tau := \mathbf{a}_\tau \mathbf{V}_\tau.$$

Using equation (3.23) and following the ideas from (3.16) and (3.17), the first two frontal slices of $\widetilde{C}_\tau$ have the form

$$\left[\begin{array}{cc|c} 1 & 0 & \\ 0 & 0 & \mathbf{0} \\ \hline \multicolumn{2}{c|}{\mathbf{0}} & \mathbf{0} \end{array}\right], \quad \left[\begin{array}{cc|c} 0 & 1 & \\ 1 & 0 & \mathbf{0} \\ \hline \multicolumn{2}{c|}{\mathbf{0}} & \mathbf{S}_{\tau_1,\tau_2} \end{array}\right].$$

We now want to determine the remaining slices similar as in (3.18). For this, first, define the matrix

$$\widetilde{\mathbf{M}} = \left[\mathrm{vec}(\mathbf{u}_3\mathbf{u}_1^\top),\ \ldots, \mathrm{vec}(\mathbf{u}_{2+k_{\tau_1}}\mathbf{u}_1^\top), \mathrm{vec}(\mathbf{u}_1\mathbf{u}_3^\top), \ldots, \mathrm{vec}(\mathbf{u}_1\mathbf{u}_{2+k_{\tau_2}}^\top)\right] \in \mathbb{C}^{k_{\tau_1}k_{\tau_2}\times(k_{\tau_1}+k_{\tau_2})},$$

where $\mathbf{u}_i$ again denotes the ith unit vector of appropriate length, with one at entry i and zeros everywhere else. This definition guarantees that

$$[\mathbf{e}_{\tau_2} \otimes \tilde{\mathbf{a}}_{\tau_1} \mid \tilde{\mathbf{a}}_{\tau_2} \otimes \mathbf{e}_{\tau_1}] = \left(\widetilde{\mathbf{U}}_{\tau_2} \otimes \widetilde{\mathbf{U}}_{\tau_1}\right)\widetilde{\mathbf{M}} \in \mathbb{C}^{n_\tau^2 \times (k_{\tau_1}+k_{\tau_2})}. \tag{3.24}$$

Thus, it encodes all single site interactions which need to be transferred to the level above. Using then the nestedness property of the HSS basis matrices, we obtain for $\tilde{\mathbf{a}}_\tau$

$$\begin{aligned} \tilde{\mathbf{a}}_\tau = \mathbf{a}_\tau\mathbf{V}_\tau &= [\mathbf{e}_{\tau_2} \otimes \mathbf{a}_{\tau_1} \mid \mathbf{a}_{\tau_2} \otimes \mathbf{e}_{\tau_1}] \begin{bmatrix} \mathbf{V}_{\tau_1} & 0 \\ 0 & \mathbf{V}_{\tau_2} \end{bmatrix} \mathbf{R}_\tau \\ &= [\mathbf{e}_{\tau_2} \otimes \tilde{\mathbf{a}}_{\tau_1} \mid \tilde{\mathbf{a}}_{\tau_2} \otimes \mathbf{e}_{\tau_1}]\,\mathbf{R}_\tau \\ &= \left(\widetilde{\mathbf{U}}_{\tau_2} \otimes \widetilde{\mathbf{U}}_{\tau_1}\right)\widetilde{\mathbf{M}}\mathbf{R}_\tau \in \mathbb{C}^{n_\tau^2 \times k_\tau}. \end{aligned}$$

Denote by $(\widetilde{\mathbf{MR}}_\tau)_i$ the matricization of the ith column of $\widetilde{\mathbf{MR}}_\tau$. Then we set the remaining k_τ slices of $\widetilde{C}_\tau$ to

$$\left[\begin{array}{cc|c} 0 & 0 & \\ 0 & 0 & \mathbf{0} \\ \hline & & \\ \mathbf{0} & & \left(\widetilde{\mathbf{MR}}_\tau\right)_i \end{array}\right] \qquad \text{for } i = 1, \dots, k_\tau \,.$$

Note that the slices with leaves as a subtree and at the root tensor have to be adjusted in the same way as for the unstructured case, cf. section 3.2. This concludes the construction of the TTNO from an HSS decomposition. The tree ranks of this construction match those of Theroem 3.2.

If the HSS ranks $k_\tau < d_\tau$, we obtain a more compact representation of the Hamiltonian in tree tensor network format. This construction involves no approximations such that the representation is still exact. Similar to the situation of TTNOs for arbitrary trees from 3.2.2 the above construction extends to this setting, by employing a suitable generalization of the HSS decomposition. However, it appears that no software for this setting is readily available. In stark contrast, multiple packages exists for the HSS binary decomposition, e.g. [MRK20].

3.3.3 Approximation by an HSS matrix

By the construction of a TTNO via the HSS decomposition from the previous section, we obtain an exact representation of an operator in tree tensor network format. For simple interaction matrices $\boldsymbol{\beta}$, i.e. next-neighbor interactions, the only nonzero entries are $\beta(i, i+1)$, for $i = 1, \dots, d-1$. In this scenario, the HSS block row $\boldsymbol{\beta}_s(\tau, \bar{\tau} \setminus \tau)$ is bounded by two, which by theorem 3.2 enables a TTNO of tree ranks at most 4. This coincides with the literature, c.f. [Tob12, Example 3.8]. However, for complicated long-range Hamiltonians, the HSS decomposition of $\boldsymbol{\beta}$ is not low-rank anymore, but often can be well approximated by such a matrix. Fortunately, the HSS framework provides a simple way to obtain an HSS decomposition with small ranks.

Fix a tolerance $\epsilon > 0$. The approximation is closely related to the low-rank approximation of the HSS block rows $\boldsymbol{\beta}_s(\tau, \bar{\tau} \setminus \tau)$. We choose k such that

$$\sigma_{k+1}(\boldsymbol{\beta}_s(\tau, \bar{\tau} \setminus \tau)) \leq \epsilon \left\|\boldsymbol{\beta}_s(\tau, \bar{\tau} \setminus \tau)\right\|_2 , \tag{3.25}$$

where σ_{k+1} denotes the $(k+1)$th singular value. The property above is known to hold for $k = \mathcal{O}(\log(d/\epsilon))$ for long-range interactions usually considered in the literature, including Coulomb interactions, see [BH09, LT21].

$\boldsymbol{\beta}_k$ is then called an (ϵ, k)-HSS matrix if every HSS block row and column of $\boldsymbol{\beta}$ admits an ϵ-approximation of at most rank k. The matrix $\boldsymbol{\beta}_k$ can be constructed by applying the SVD-based procedures from [XXCB14, MRK20] to the symmetrized interaction matrix $\boldsymbol{\beta}_s$. Each off-diagonal block of $\boldsymbol{\beta}_k$ can be written as

$$\boldsymbol{\beta}_k(\tau_1, \tau_2) = \mathbf{V}_{\tau_1,\epsilon}\mathbf{S}_{\tau_1,\tau_2,\epsilon}\mathbf{V}^*_{\tau_2,\epsilon}, \qquad \mathbf{V}_{\tau_1,\epsilon} \in \mathbb{C}^{d_{\tau_1}\times k_{\tau_1}},\ \mathbf{S}_{\tau_1,\tau_2,\epsilon} \in \mathbb{C}^{k_{\tau_1}\times k_{\tau_2}},\ \mathbf{V}_{\tau_2,\epsilon} \in \mathbb{C}^{d_{\tau_2}\times k_{\tau_1}},$$

with reduced HSS ranks k_{τ_i} for all $\tau \leq \bar{\tau}$.

The error of this rank-k approximation to the full $\boldsymbol{\beta}$ is proportional to ϵ, which follows by applying the next lemma to the symmetrized interaction matrix $\boldsymbol{\beta}_s$. The lemma and its proof can be found in [XXCB14, Corollary 4.3], giving us the bound.

Lemma 3.3

Let $\boldsymbol{\beta} \in \mathbb{C}^{d\times d}$ be a strictly upper triangular interaction matrix such that (3.25) is satisfied for a binary dimension tree $\bar{\tau}$ and some $\epsilon > 0$. Then there exists a strictly upper triangular matrix $\boldsymbol{\beta}_k \in \mathbb{C}^{d\times d}$ in HSS decomposition (3.19)–(3.20) of HSS rank k such that

$$\|\boldsymbol{\beta} - \boldsymbol{\beta}_k\|_F \leq Ch(\bar{\tau})\sqrt{k}\|\boldsymbol{\beta}\|_{\mathrm{F}} \cdot \epsilon\,.$$

is satisfied for some constant C, where $h(\bar{\tau})$ denotes the height of $\bar{\tau}$.

We can use the construction described in subsection 3.3.2 using the approximated HSS matrix $\boldsymbol{\beta}_k$ to obtain an approximated TTNO decomposition of the corresponding Hamiltonian $\widehat{H}_k$. The next theorem shows that $\widehat{H}_k$ is ϵ-close to the exact Hamiltonian $\widehat{H}$ in spectral norm. It can be found in [CKS24, Theorem 4.5].

Theorem 3.3

Under the setting and assumptions of Lemma 3.3, let $\widehat{H}$ be the linear operator defined by (3.5). Then

$$\widehat{H}_k = \sum_{i<j}^{d} \beta_k(i,j) \cdot \mathbf{A}^{(i)}\mathbf{A}^{(j)} \in \mathbb{C}^{(n_1\cdots n_d)\times(n_1\cdots n_d)},$$

has TTNO rank $2 + k$ and satisfies the error bound

$$\left\|\widehat{H} - \widehat{H}_k\right\|_2 \leq Ch(\bar{\tau})\sqrt{k}\|\boldsymbol{\beta}\|_{\mathrm{F}}\Big(\sum_{i<j}^{d}\|\mathbf{A}_i\|_2^2\|\mathbf{A}_j\|_2^2\Big)^{1/2} \cdot \epsilon.$$

Proof. Using the triangular inequality and the Cauchy-Schwartz inequality, we obtain the bound

$$\begin{aligned}\left\|\widehat{H}-\widehat{H}_k\right\|_2 &= \Big\|\sum_{i<j}^{d}(\beta(i,j)-\beta_k(i,j))\mathbf{A}^{(i)}\mathbf{A}^{(j)}\Big\|_2 \\ &\leq \sum_{i<j}^{d}|\beta(i,j)-\beta_k(i,j)|\cdot\|\mathbf{A}^{(i)}\|_2\|\mathbf{A}^{(j)}\|_2 \\ &= \sum_{i<j}^{d}|\beta(i,j)-\beta_k(i,j)|\cdot\|\mathbf{A}_i\|_2\|\mathbf{A}_j\|_2 \\ &\leq \|\boldsymbol{\beta}-\boldsymbol{\beta}_k\|_F\Big(\sum_{i<j}^{d}\|\mathbf{A}_i\|_2^2\|\mathbf{A}_j\|_2^2\Big)^{1/2}.\end{aligned}$$

We apply lemma 3.3 to bound the frist term by

$$\|\boldsymbol{\beta}-\boldsymbol{\beta}_k\| \leq Ch(\bar{\tau})\sqrt{k}\|\boldsymbol{\beta}\|_{\mathrm{F}}\cdot\epsilon,$$

which gives the stated error bound. Using theorem 3.2, we obtain that the maximal tree rank equals $k+2$, which concludes the proof. □

Note that the double sum $\sum_{i<j}^{d}\|\mathbf{A}_i\|_2^2\|\mathbf{A}_j\|_2^2$ appearing on the right side of the error bound can become large if the spectral norm of the matrices $\mathbf{A}_i$ is large. However, in the regime of quantum spin systems, where $\mathbf{A}_i$ are usually Pauli matrices or identities, we have $\|\mathbf{A}_i\|_2 = 1$ and hence the double sum only contributes with a factor of the order $\mathcal{O}(d)$.

3.4 Numerical examples

In this section we verify the theoretical results from above by applying both strategies to construct tree tensor network operators to several quantum spin systems.

3.4.1 Closed quantum system operators

We consider a long-range Hamiltonian for unitary dynamics for d spin-$\frac{1}{2}$ particles

$$\mathcal{H} = \Omega\sum_{k=1}^{d}\boldsymbol{\sigma}_x^{(k)} + \Delta\sum_{k=1}^{d}\boldsymbol{n}^{(k)} + \nu\sum_{k<h}\frac{1}{|h-k|^{\alpha}}\boldsymbol{n}^{(k)}\boldsymbol{n}^{(h)}, \tag{3.26}$$

where the appearing matrices are defined as

$$\boldsymbol{\sigma}_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \boldsymbol{n} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

$\boldsymbol{\sigma}_x$ denotes the first Pauli matrix, while $\boldsymbol{n}$ denotes the projector onto the excited state. The notation $\boldsymbol{\sigma}^{(k)} = \mathbf{I} \otimes \cdots \otimes \mathbf{I} \otimes \boldsymbol{\sigma} \otimes \mathbf{I} \otimes \cdots \otimes \mathbf{I}$ denotes the action of the matrix $\boldsymbol{\sigma}$ on the kth site. Ω, Δ, V are real numbers and $\alpha \geq 0$ - the parameters of the system. The first two terms describe a driving term, e.g. an excitation by a laser with Rabi frequency Ω and detuning Δ. The ν describes the strength of the interaction between the particles, while α is an interaction exponent. For a more detailed model description, see [SWM10]. The parameter α allows to interpolate between different regimes:

- $\alpha = 0$ encodes an all-to-all interaction;
- $\alpha = \infty$ encodes nearest-neighbor interactions;
- $0 < \alpha < \infty$ encodes long-range interactions.

Some examples of the α parameter choice are given by $\alpha = 1$ which encode Coulomb interactions, $\alpha = 3$ dipole-dipole interactions or $\alpha = 6$ van der Waals interactions [SWM10].

The TTNO corresponding to the full Hamiltonian (3.26) is constructed by two TTNO's - one for the diagonal part and one for the interaction part respectively. Note that single sums can be constructed as a rank 2 TTNO, where one rank encodes the identity. As the identity is already encoded in the double sum part, we only get one additional rank in the full TTNO, stemming form the diagonal part.

First, we want to verify that both above strategies are equivalent. We construct the Hamiltonian (3.26) by the unstructured strategy from section 3.2 and the HSS strategy from the previous section 3.3 for different number of sites d. We compute the difference in Frobenius norm as explained in (2.8). To obtain comparable results for all d, we scale the error with the norm of the TTNO resulting from the unstructured strategy, i.e. we compute $\frac{\|H_{\text{unstruct}} - H_{\text{HSS}}\|}{\|H_{\text{unstruct}}\|}$. In figure 3.2 we see that the difference is of the order of 10^{-12}, which is the default tolerance in the hm-toolbox used for computing the HSS decomposition of the interaction matrix [MRK20]. Hence, both strategies are equivalent. Further, the TTNO rank satisfies the theoretical bound, with an additional rank arising from the Laplacian-like part of the operator, i.e. the single sums in (3.26).

ϵ-dependence

From figure 3.2 we observe that the error of the HSS construction of the TTNO is

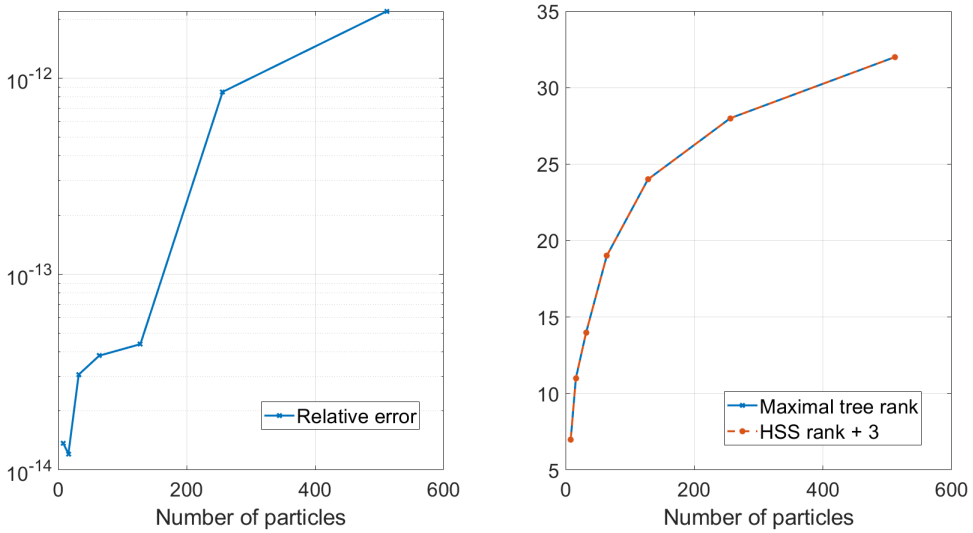

FIGURE 3.2: Long-range tree tensor network operator of a closed system with parameters $\Omega = 3$, $\Delta = -2$, $\nu = 2$ and $\alpha = 1$. Left: Relative error of the TTNO vs the number of particles. Right: Maximal tree rank (solid line) and expected tree rank (dashed line) vs the number of particles.

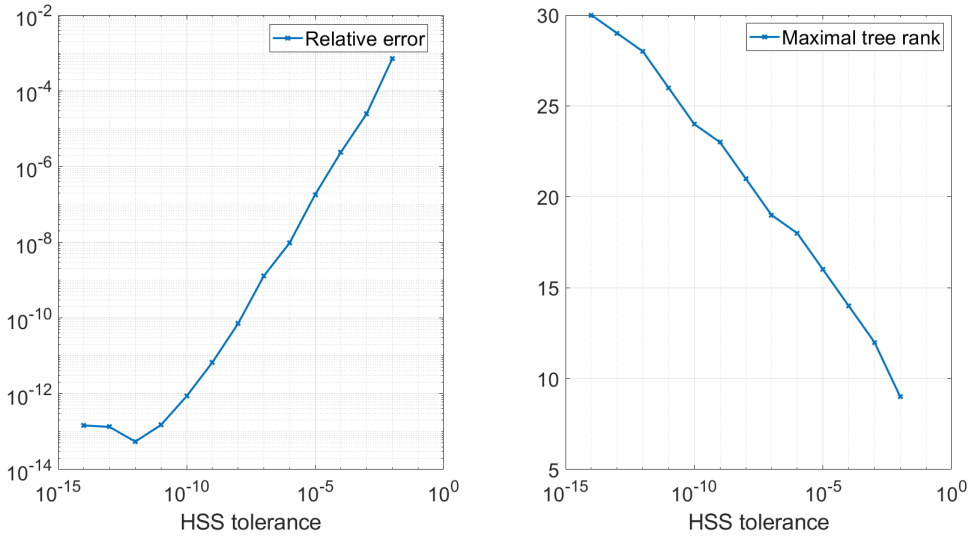

FIGURE 3.3: Long-range tree tensor network operator of a closed system with parameters $\Omega = 3$, $\Delta = -2$, $\nu = 2$, $\alpha = 1$ and $d = 256$. Left: Relative error of the TTNO vs HSS tolerance. Right: Maximal tree rank vs HSS tolerance.

bounded by the HSS tolerance ϵ. Thus, we want to study the influence of the HSS tolerance ϵ in more detail. We fix $\alpha = 1$ and $d = 256$ and compute the error for different HSS tolerances ϵ. In figure 3.3 we observe a linear dependence between the relative error and the HSS tolerance. With increasing tolerance, we further observe a decreasing maximal tree rank. Analogous behaviour is obtained for different parameters and number of particles. Hence, we can use this strategy to find approximations to a Hamiltonian.

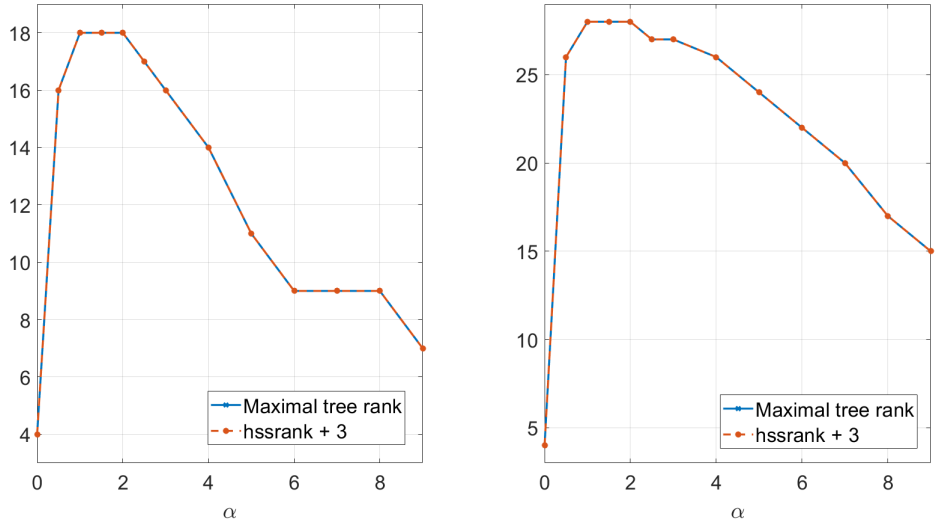

FIGURE 3.4: Long-range tree tensor network operator of a closed system with parameters $\Omega = 3$, $\Delta = -2$, $\nu = 2$ and $d = 256$. Left: Tree ranks vs different α for HSS tolerance $\epsilon = 10^{-6}$. Right: Tree ranks vs different α for HSS tolerance $\epsilon = 10^{-12}$.

Parameter study

Finally, we want to study in more detail the influence of the parameter α. We have already discussed in subsection 3.3.3 that for $\alpha = \infty$ there exists a TTNO representation of maximal rank 4. We vary now α across different interaction regimes and check for the complexity of the resulting TTNO. For $\alpha \geq 1$, an increase in α leads to a decrease in the maximal tree ranks, see figure 3.4. By this result, Coulomb interactions, i.e. values around $\alpha = 1$, are computationally the most challenging regime for this system for different approximation values ϵ.

3.4.2 Open quantum system operators

The interaction of a quantum system with its environment makes this an open quantum system. The aim is now to find an approximation to a one-dimensional quantum system of d distinguishable spin-$\frac{1}{2}$ particles. The (vectorized) quantum state density matrix $\rho(t)$ now evolves according to the quantum master equation

$$\dot{\rho}(t) = \mathcal{L}[\rho(t)] := -i[H, \rho(t)] + \mathcal{D}[\rho(t)] \tag{3.27}$$

where H is a scaled version of the Hamiltonian (3.26) from above, $[\cdot, \cdot]$ the commutator bracket and $\mathcal{D}$ the decay operator (3.29). The operator $\mathcal{L}$ is called the Lindblad operator, see [BP02] for details and note the original work of Lindblad [Lin76]. For the simulations

we use

$$H = \Omega \sum_{k=1}^{d} \boldsymbol{\sigma}_x^{(k)} + \Delta \sum_{k=1}^{d} \boldsymbol{n}^{(k)} + \frac{\nu}{c_\alpha} \sum_{k \neq h}^{d} \frac{\boldsymbol{n}^{(k)} \boldsymbol{n}^{(h)}}{|k-h|^\alpha}, \tag{3.28}$$

$$\mathcal{D} = \sum_{k=1}^{d} \left[\boldsymbol{J} \otimes (\boldsymbol{J}^*)^\top - \frac{1}{2} \boldsymbol{J}^* \boldsymbol{J} \otimes \mathbf{I} - \frac{1}{2} \mathbf{I} \otimes (\boldsymbol{J}^* \boldsymbol{J})^\top \right]^{(k)}, \tag{3.29}$$

where

- $\boldsymbol{n} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \frac{\sigma_z + \mathbf{I}}{2}$ is again the projector onto the excited state acting on the kth particle,
- $c_\alpha = \sum_{k=1}^{d} \frac{1}{k^\alpha}$, allows to keep the interaction extensive.
- $\boldsymbol{J} = \sqrt{\gamma} \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$ are the so-called jump-operators, which encode how the external environment affects the dynamics. The $\boldsymbol{J}$ matrices describe a local transition from an excited state to the ground state, with a local decay rate of γ.

For a detailed description of this model, we refer to [SLC+24] and the references therein. Inserting now (3.28) and (3.29) into the Lindblad equation from (3.27) and inserting the commutator bracket, we obtain

$$\begin{aligned} \mathcal{L} = {} & \Omega \sum_{k=1}^{d} \left[-i \boldsymbol{\sigma}_x \otimes \mathbf{I} + i \mathbf{I} \otimes \boldsymbol{\sigma}_x^\top \right]^{(k)} + \Delta \sum_{k=1}^{d} \left[-i \boldsymbol{n} \otimes \mathbf{I} + i \mathbf{I} \otimes \boldsymbol{n}^\top \right]^{(k)} \\ & + \sum_{k=1}^{d} \left[\boldsymbol{J} \otimes (\boldsymbol{J}^*)^\top - \frac{1}{2} \boldsymbol{J}^* \boldsymbol{J} \otimes \mathbf{I} - \frac{1}{2} \mathbf{I} \otimes (\boldsymbol{J}^* \boldsymbol{J})^\top \right]^{(k)} \\ & + \sum_{k \neq h} \frac{-i\nu}{c_\alpha |k-h|^\alpha} \left[\boldsymbol{n} \otimes \mathbf{I} \right]^{(k)} \left[\boldsymbol{n} \otimes \mathbf{I} \right]^{(h)} + \sum_{k \neq h} \frac{i\nu}{c_\alpha |k-h|^\alpha} \left[\mathbf{I} \otimes \boldsymbol{n}^\top \right]^{(k)} \left[\mathbf{I} \otimes \boldsymbol{n}^\top \right]^{(h)} \end{aligned}$$

As for the closed system, we first check the equivalence of both strategies to construct the TTNO. Note that two double sums appear in the Lindblad operator because of the commutator bracket. Hence, we use the unstructured and HSS construction twice, once for each double sum. In figure 3.5 we again see that the model allows for a low-rank TTNO representation. However, the maximal tree rank is larger than for the closed system, due to the fact that we have two double sums instead of only one. The relative error is again of the order of the chosen HSS tolerance.

ϵ-dependence

The linear behaviour of the relative error with respect to the HSS tolerance is also seen for the open system, see figure 3.6. It is noteworthy that the relative error stays constant

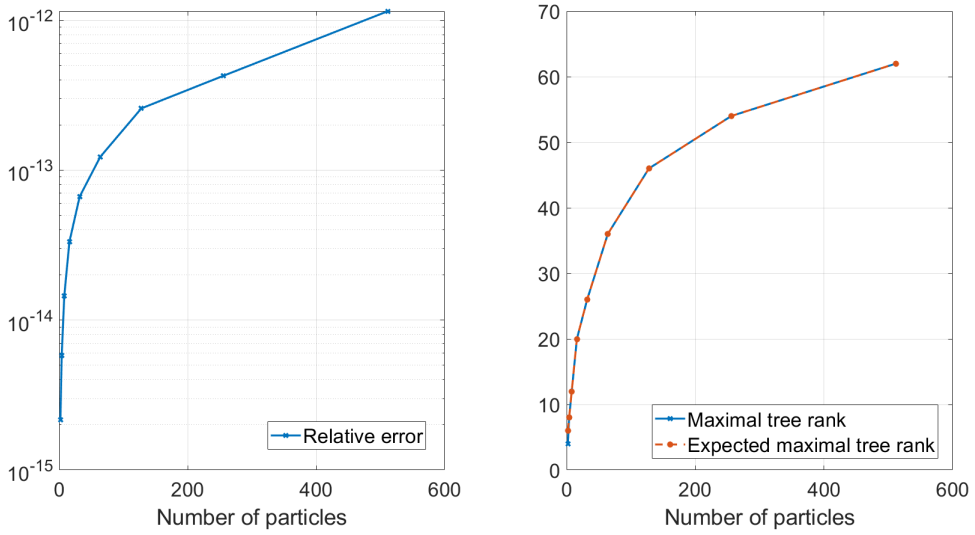

FIGURE 3.5: Long-range tree tensor network operator of an open system with parameters $\Omega = 3$, $\Delta = -2$, $\nu = 2$, $\gamma = 1$ and $\alpha = 1$. Left: Relative error of the TTNO vs the number of particles. Right: Maximal tree rank (solid line) and expected tree rank (dashed line) vs the number of particles.

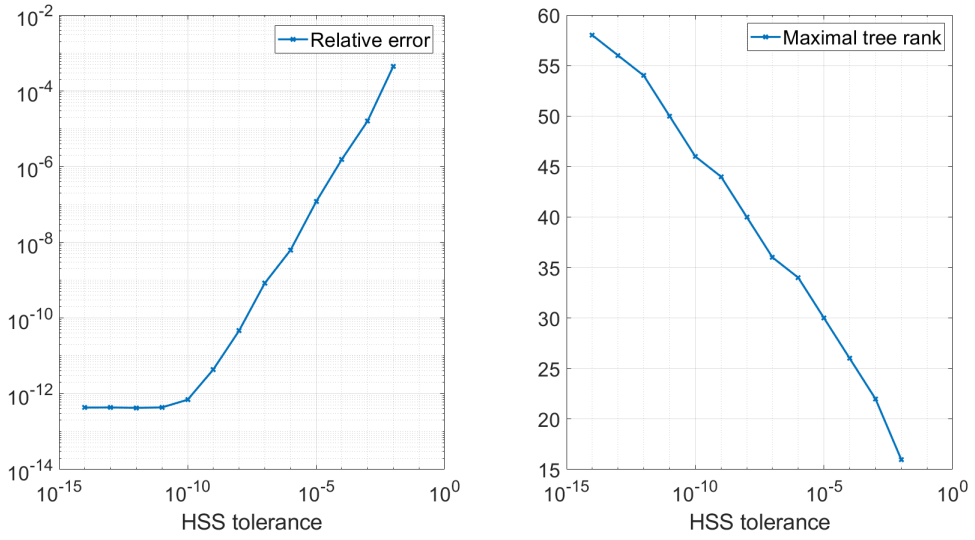

FIGURE 3.6: Long-range tree tensor network operator of an open system with parameters $\Omega = 3$, $\Delta = -2$, $\nu = 2$, $\gamma = 1$, $\alpha = 1$ and $d = 256$. Left: Relative error of the TTNO vs HSS tolerance. Right: Maximal tree rank vs HSS tolerance.

up to $\epsilon = 10^{-10}$, while the maximal tree rank already reduces from 58 ($\epsilon = 10^{-14}$) to 46 ($\epsilon = 10^{-10}$). This indicates that a mild truncation/approximation can lead to significantly reduced ranks while only a very mild loss in accuracy is observed.

Parameter study

As in the last subsection, we perform a parameter study of α to check the complexity of the corresponding TTNO. In figure 3.7 we see that in the open setting, Coulomb

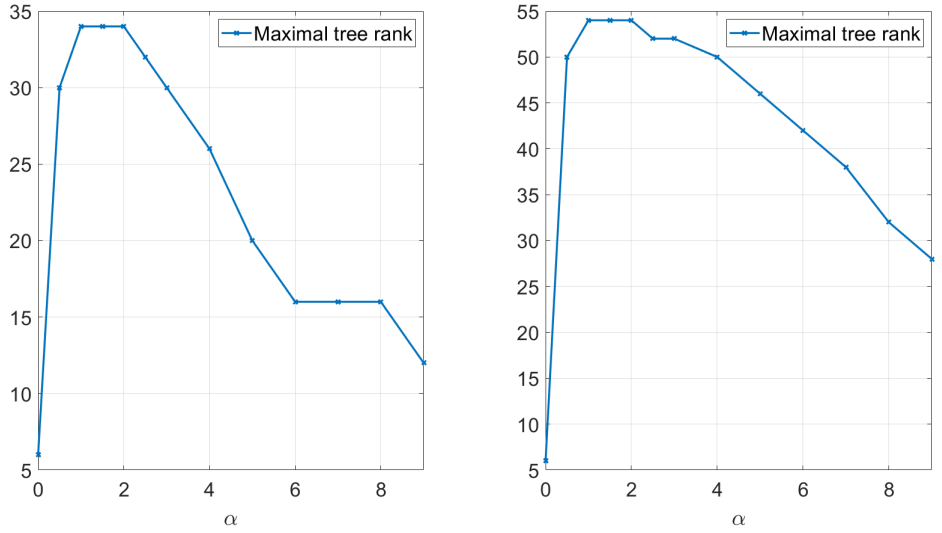

FIGURE 3.7: Long-range tree tensor network operator of an open system with parameters $\Omega = 3$, $\Delta = -2$, $\nu = 2$, $\gamma = 1$ and $d = 256$. Left: Tree ranks vs different α for HSS tolerance $\epsilon = 10^{-6}$. Right: Tree ranks vs different α for HSS tolerance $\epsilon = 10^{-12}$.

interactions are again the computationally most challenging regime. However, choosing larger HSS tolerances can be used to find approximations of lower rank.

3.4.3 Balanced binary tree vs. Tensor train

An HSS decomposition can be performed for any underlying binary tree. Hence, we gain the flexibility to explore how different tree structures influence the tree ranks and memory complexity of the corresponding TTNO. We will focus on balanced binary trees and unbalanced binary trees, recall figure 3.1 for a graphical representation of those. The latter represents the well-known tensor format of a tensor train/matrix product state. For the closed and open system from above, we construct the corresponding TTNOs in both tree formats. In figure 3.8 for the closed and figure 3.9 for the open system, we see that the maximal tree rank is lower in the tensor train format. However, this is only one measure of the overall complexity. If one considers the total memory footprint of the TTNOs, we see that the balanced binary tree is memory-wise equal or more compact than the tensor train format. The tree ranks in the balanced binary tree decrease faster than in the tensor train format, which is most likely due to the logarithmic scaling between the sites in the balanced binary tree. This indicates that tree tensor networks based on balanced binary trees are suited better for simulations with long-range interacting quantum spin systems. For the moment this holds only true on the level of the tree tensor network operators. Later in chapter 6 we will see that a similar result holds true for the time evolution of a quantum state with the same operators.

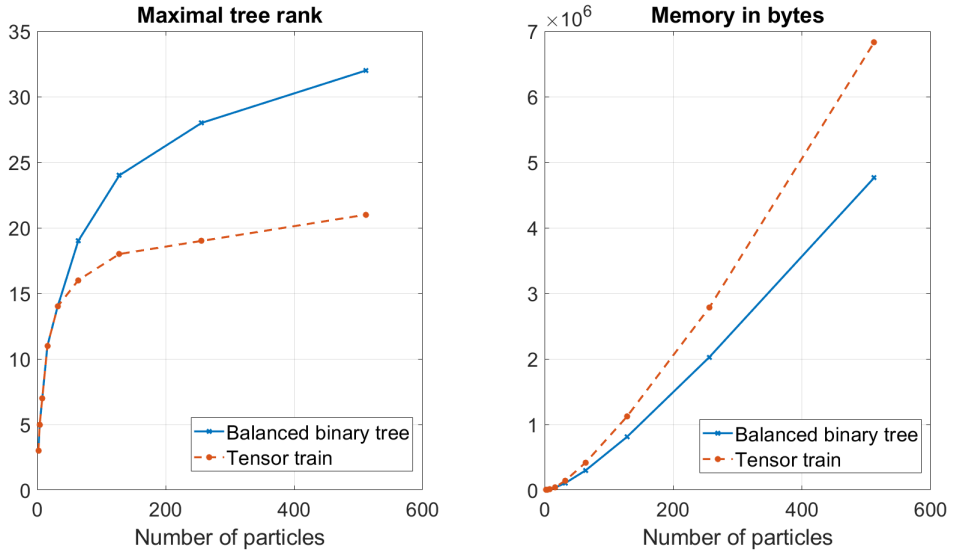

FIGURE 3.8: Long-range tree tensor network operator of a unitary system with parameters $\Omega = 3$, $\Delta = -2$, $\nu = 2$ and $\alpha = 1$. Left: Maximal tree ranks of the TTNO for a balanced binary tree (solid line) and an unbalanced binary tree vs the number of particles. Right: Memory footprint in bytes of the TTNO for a balanced binary tree (solid line) and an unbalanced binary tree vs number of particles

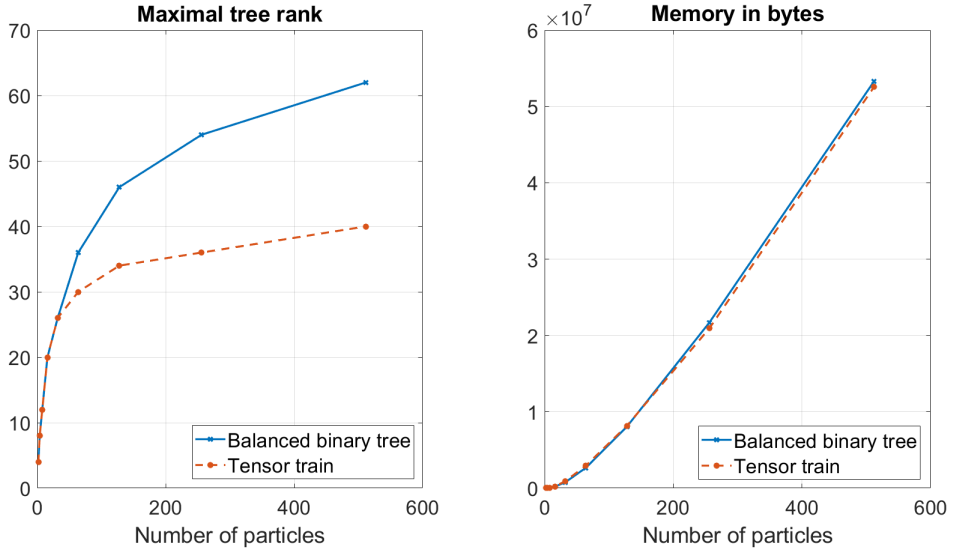

FIGURE 3.9: Long-range tree tensor network operator of an open system with parameters $\Omega = 3$, $\Delta = -2$, $\nu = 2$, $\gamma = 1$ and $\alpha = 1$. Left: Maximal tree ranks of the TTNO for a balanced binary tree (solid line) and an unbalanced binary tree vs the number of particles. Right: Memory footprint in bytes of the TTNO for a balanced binary tree (solid line) and an unbalanced binary tree vs number of particles

Chapter 4

Rank-adaptive BUG integrator for Tree Tensor Networks

This chapter is mainly based on the work "Rank-adaptive time integration of tree tensor networks" by Gianluca Ceruti, Christian Lubich and the author [CLS23].

4.1 Recap: Rank-adaptive integrator for Tucker tensors

In this section we want to recap the rank-adaptive integrator for Tucker tensors of [CKL22]. Let $Y^0 \in \mathcal{M}_r$ be an element from the manifold of order d tensors with multilinear rank $r = (r_1^0, \dots, r_d^0)$. By (2.3) we can decompose Y^0 through a Tucker decomposition

$$Y^0 = C^0 \bigtimes_{j=1}^{d} \mathbf{U}_j^0.$$

Following the approach suggested in [CLW21, CLS23], we introduce the subflows $\Phi^{(i)}$ and Ψ which correspond to the updates of the leaves and the core tensor respectively. As we will see in the formulation of the algorithm, all subflows $\Phi^{(i)}$ can be done in parallel, i.e. we obtain

$$\widehat{Y}^1 = \Psi \circ (\Phi^{(1)}, \dots, \Phi^{(d)})(Y^0).$$

$\widehat{Y}^1$ is a Tucker tensor of multilinear rank $\widehat{r} = (\widehat{r}_1, \dots, \widehat{r}_d)$, where $\widehat{r}_j \leq 2r_j^0$. As the multilinear ranks increased in this step, we apply a rank truncation algorithm Θ with a given tolerance parameter ϑ, cf. algorithm 1. The approximation Y^1 at time t_1 can be

written schematically as

$$Y^1 = \Theta(\widehat{Y}^1) = \Theta \circ \Psi \circ (\Phi^{(1)}, \dots, \Phi^{(d)})(Y^0).$$

Algorithm 2: Rank-adaptive BUG integrator for Tucker tensors

Data: Tucker tensor $Y^0 = C^0 \times_{i=1}^d \mathbf{U}_i^0$ in factorized form of multilinear rank $(r_1^0, \dots, r_d^0)$, function $F(t, Y)$, t_0, t_1, tolerance parameter ϑ

Result: Tucker tensor $Y^1 = C^1 \times_{i=1}^d \mathbf{U}_i^1$ in factorized form of multilinear rank $(r_1^1, \dots, r_d^1)$, where $r_i^1 \le 2r_i^0$

begin

  for $i = 1 : d$ in parallel **do**

    compute $[\widehat{\mathbf{U}}_i^1, \widehat{\mathbf{M}}_i] = \Phi^{(i)}\big(Y^0, F, t_0, t_1\big)$

    `% update and augment the` i`th basis matrix`

  end

  compute $\widehat{C}^1 = \Psi\big(C^0, (\widehat{\mathbf{U}}_i^1)_{i=1}^d, (\widehat{\mathbf{M}}_i)_{i=1}^d, F, t_0, t_1\big)$

  `% augment and update the core tensor`

  set $\widehat{Y}^1 = \widehat{C}^1 \times_{i=1}^d \widehat{\mathbf{U}}_i^1$

  compute $Y^1 = \Theta\big(\widehat{Y}^1, \vartheta\big)$ `% rank truncation`

end

Algorithm 3: Subflow $\Phi^{(i)}$ (update and augment the ith basis matrix))

Data: Tucker tensor $Y^0 = C^0 \times_{j=1}^d \mathbf{U}_j^0$ of multilinear rank $(r_1, \dots, r_d)$ in factorized form, function $F(t, Y)$, t_0, t_1

Result: Updated and augmented basis matrix $\widehat{\mathbf{U}}_i^1 \in \mathbb{C}^{n_i \times \widehat{r}_i}$ (typically $\widehat{r}_i = 2r_i$) with orthonormal columns, auxiliary matrix $\widehat{\mathbf{M}}_i$

begin

  compute the QR-decomposition $\mathbf{Mat}_i(C^0)^\top = \mathbf{Q}_i^0 \mathbf{S}_i^{0,\top}$;

  solve the $n_i \times r_i$ matrix differential equation from t_0 to t_1

$$\dot{\mathbf{K}}_i(t) = \mathbf{F}_i\big(t, \mathbf{K}_i(t)\mathbf{V}_i^{0,*}\big)\mathbf{V}_i^0, \qquad \mathbf{K}_i(t_0) = \mathbf{U}_i^0 \mathbf{S}_i^0,$$

  with $\mathbf{F}_i(t, \cdot) := \mathbf{Mat}_i \circ F(t, \cdot) \circ \mathit{Ten}_i$ and $\mathbf{V}_i^{0,*} := \mathbf{Q}_i^\top \otimes_{j \ne i}^d \mathbf{U}_j^{0,\top}$;

  compute $\widehat{\mathbf{U}}_i^1$ as an orthonormal basis of the range of the $n_i \times 2r_i$ matrix $\big(\mathbf{K}_i(t_1), \mathbf{U}_i^0\big)$;

  set $\widehat{\mathbf{M}}_i = \widehat{\mathbf{U}}_i^{1,*} \mathbf{U}_i^0$.

end

Algorithm 4: Subflow Ψ (augment and update the core tensor)

Data: core tensor $C^0 \in \mathbb{C}^{r_1 \times \cdots \times r_d}$, augmented basis matrices $\widehat{\mathbf{U}}_i^1 \in \mathbb{C}^{n_i \times \widehat{r}_i}$ with orthonormal columns, auxiliary matrices $\widehat{\mathbf{M}}_i \in \mathbb{C}^{\widehat{r}_i \times r_i}$, function $F(t, Y)$, t_0, t_1

Result: core tensor $\widehat{C}^1 \in \mathbb{C}^{\widehat{r}_1 \times \cdots \times \widehat{r}_d}$

begin

solve the $\widehat{r}_1 \times \cdots \times \widehat{r}_d$ tensor differential equation from t_0 to t_1,

$$\dot{\widehat{C}}(t) = F\big(t, \widehat{C}(t) \bigtimes_{i=1}^{d} \widehat{\mathbf{U}}_i^1\big) \bigtimes_{i=1}^{d} \widehat{\mathbf{U}}_i^{1,*}, \quad \widehat{C}(t_0) = C^0 \bigtimes_{i=1}^{d} \widehat{\mathbf{M}}_i;$$

set $\widehat{C}^1 = \widehat{C}(t_1)$

end

4.2 Extended rank-adaptive integrator for Tucker tensors

We extend the algorithm from above to r-tuples of Tucker tensors. This will be a crucial ingredient for formulating the rank-adaptive BUG integrator for tree tensor networks. Consider a tensor in Tucker format of multilinear rank $r = (r_0, r_1, \cdots, r_d)$, where in the 0-dimension only a $r_0 \times r_0$ identity matrix appears, i.e.

$$Y^0 = C^0 \times_0 \mathbf{I} \bigtimes_{j=1}^{d} \mathbf{U}_j^0.$$

As described in previous subsection, we obtain the approximation Y^1 by

$$Y^1 = \Theta \circ \Psi \circ (\Phi^{(d)}, \ldots, \Phi^{(1)}, \Phi^{(0)})(Y^0).$$

We will see in the following lemma that with an appropriate orthogonalization, the subflow $\Phi^{(0)}$ becomes trivial, i.e.

$$Y^1 = \Theta \circ \Psi \circ (\Phi^{(d)}, \ldots, \Phi^{(1)})(Y^0).$$

Lemma 4.1

With the appropriate choice of orthogonalization, the action of the subflow $\Phi^{(0)}$ on Y^0 becomes trivial, i.e.

$$\Phi^{(0)}(Y^0) = [\widehat{\mathbf{U}}_0^1, \widehat{\mathbf{M}}_0] = Y^0.$$

Proof.
In the subflow $\Phi^{(0)}$ we solve the $r_0 \times r_0$ matrix differential equation

$$\dot{\mathbf{K}}_0(t) = \mathbf{F}_i\left(t, \mathbf{K}_0(t)\mathbf{V}_0^{0,*}\right)\mathbf{V}_0^0, \quad \mathbf{K}_0(t_0) = \mathbf{U}_0^0\mathbf{S}_0^0.$$

We define $\mathbf{K}_0^1 = \mathbf{K}_0(t_1)$ as the solution at time t_1. Obviously the matrix

$$(\mathbf{K}_0^1, \mathbf{U}_0^0) = (\mathbf{K}_0^1, \mathbf{I}) \in \mathbb{C}^{r_0 \times 2r_0}$$

has exactly rank r_0. Therefore, the columns of $\widehat{\mathbf{U}}_0^1 = \mathbf{I}$ form an orthonormal basis of the range of this matrix. Moreover, $\widehat{\mathbf{M}}_0 = \widehat{\mathbf{U}}_0^{1,*}\mathbf{U}_0^0 = \mathbf{I}^*\mathbf{I} = \mathbf{I}$. □

4.3 Rank-adaptive BUG integrator for Tree Tensor Networks

At time t_0, consider now a tree tensor network

$$Y_\tau^0 = C_\tau^0 \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^{m} \mathbf{U}_{\tau_i}^0,$$

with associated tree $\tau = (\tau_1, \ldots, \tau_m)$. Seeing this as an extended Tucker tensor we can apply the extended rank-adaptive Tucker integrator with a given function F_τ. We then obtain

$$\widehat{Y}_\tau^1 = \Psi_\tau \circ (\Phi_\tau^{(m)}, \ldots, \Phi_\tau^{(1)})(Y_\tau^0).$$

The subscript τ indicates that the update was done with respect to the tree τ and the associated function F_τ. However, for computational efficiency, one does not want to compute the matrices $\mathbf{U}_{\tau_i}^0$, unless they are basis matrices. To update now the (possibly inaccessible) matrices $\mathbf{U}_{\tau_i}^0$, which corresponds to the subflow $\Phi_\tau^{(i)}$, we differentiate between two cases:

1. If $\tau_i = l$ is a leaf, we directly apply the subflow $\Phi_\tau^{(i)}$ to update and augment the basis matrix.
2. If τ_i is not a leaf, we apply the algorithm recursively. For that we use the reduced initial data $Y_{\tau_i}^0 = \pi_{\tau,i}^\dagger(Y_\tau^0)$ and the reduced function $F_{\tau_i} = \pi_{\tau,i}^\dagger \circ F_\tau \circ \pi_{\tau,i}$, which will be discussed in section 4.5.

We can now formulate the rank-adaptive integrator for tree tensor networks. The algorithm is applied recursively from the root to the leaves and consists of the update and

augmenting steps (algorithm 5 with algorithm 6 and 7), followed by a final truncation (algorithm 1). The difference to the extended Tucker integrator is that the subflow $\Phi_\tau^{(i)}$ is applied recursively from the leaves to the root. This is an approximative subflow as the differential equation is only solved approximately by recurrence unless τ_i is a leaf [CLW21].

The $\widehat{\mathbf{U}}^1_{\tau_i}$ in algorithm 5 corresponds to the updated and augmented basis matrix. Typically, the rank is doubled, i.e. $\widehat{r}_{\tau_i} = 2r^0_{\tau_i}$. Note that $\widehat{\mathbf{U}}^1_{\tau_i}$ is again only stored in factorized form. The tensor $\widehat{C}^1_\tau \in \mathbb{C}^{r_\tau \times \widehat{r}_{\tau_1} \times \cdots \times \widehat{r}_{\tau_m}}$ is the updated and augmented core tensor, while $\widehat{C}^0_\tau \in \mathbb{C}^{r_\tau \times \widehat{r}_{\tau_1} \times \cdots \times \widehat{r}_{\tau_m}}$ is the augmented initial data.

Algorithm 5: Rank-augmenting TTN integrator

Data: tree $\tau = (\tau_1, \ldots, \tau_m)$, TTN $Y^0_\tau = C^0_\tau \times_0 \mathbf{I}_\tau \times_{i=1}^m \mathbf{U}^0_{\tau_i}$ in factorized form of tree rank $(r^0_\sigma)_{\sigma \le \tau}$, function $F_\tau(t, Y_\tau)$, t_0, t_1

Result: TTN $\widehat{Y}^1_\tau = \widehat{C}^1_\tau \times_0 \mathbf{I}_\tau \times_{i=1}^d \widehat{\mathbf{U}}^1_{\tau_i}$ in factorized form of augmented tree rank $(\widehat{r}_\sigma)_{\sigma \le \tau}$ with $\widehat{r}_\sigma \le 2r^0_\sigma$, augmented connection tensor $\widehat{C}^0_\tau$

begin

  for $i = 1 : m$ in parallel **do**

    compute $[\widehat{\mathbf{U}}^1_{\tau_i}, \widehat{\mathbf{M}}_{\tau_i}] = \Phi^{(i)}_\tau\big(Y^0_\tau, F_\tau, t_0, t_1\big)$

    `% update and augment the basis matrix for subtree` τ_i`, see algorithm 6`

  end

  compute $[\widehat{C}^1_\tau, \widehat{C}^0_\tau] = \Psi_\tau\left(C^0_\tau, (\widehat{\mathbf{U}}^1_{\tau_i})_{i=1}^m, (\widehat{\mathbf{M}}_{\tau_i})_{i=1}^m, F_\tau, t_0, t_1\right)$

  `% augment and update the connection tensor, see algorithm 7`

  set $\widehat{Y}^1_\tau = \widehat{C}^1_\tau \times_0 \mathbf{I}_\tau \times_{i=1}^m \widehat{\mathbf{U}}^1_{\tau_i}$

end

Algorithm 6: Subflow $\Phi_\tau^{(i)}$ (update and augment a basis matrix)

Data: tree $\tau = (\tau_1, \dots, \tau_m)$, TTN $Y_\tau^0 = C_\tau^0 \times_0 \mathbf{I}_\tau \times_{i=1}^m \mathbf{U}_{\tau_i}^0$ in factorized form of tree rank $(r_\sigma^0)_{\sigma \le \tau}$, with $\mathbf{U}_{\tau_i}^0 = \mathbf{Mat}_0(X_{\tau_i}^0)^\top \in \mathbb{C}^{n_{\tau_i} \times r_{\tau_i}^0}$, function $F_\tau(t, Y_\tau)$, t_0, t_1

Result: $\widehat{\mathbf{U}}_{\tau_i}^1 = \mathbf{Mat}_0(\widehat{X}_{\tau_i}^1)^\top \in \mathbb{C}^{n_{\tau_i} \times \widehat{r}_{\tau_i}}$ (typically $\widehat{r}_{\tau_i} = 2r_{\tau_i}^0$) in factorized form, auxiliary matrix $\widehat{\mathbf{M}}_{\tau_i} \in \mathbb{C}^{\widehat{r}_{\tau_i} \times r_{\tau_i}^0}$

begin

compute a QR-decomposition $\mathbf{Mat}_i(C_\tau^0)^\top = \mathbf{Q}_{\tau_i}^0 \mathbf{S}_{\tau_i}^{0,\top}$;

set $Y_{\tau_i}^0 = X_{\tau_i}^0 \times_0 \mathbf{S}_{\tau_i}^{0,\top}$

if $\tau_i = l$ is a leaf **then**

solve the $n_l \times r_l^0$ matrix differential equation

$$\dot{Y}_l(t) = F_l(t, Y_l(t)), \quad Y_l(t_0) = Y_l^0 \in \mathbb{C}^{r_l^0 \times n_l};$$

compute $\widehat{\mathbf{U}}_l^1 \in \mathbb{C}^{n_l \times \widehat{r}_l}$ with $\widehat{r}_l \le 2r_l^0$ as an orthonormal basis of the range of the $n_l \times 2r_l^0$ matrix $\big(Y_l(t_1)^\top, \mathbf{U}_l^0\big)$;

set $\widehat{\mathbf{M}}_l = \widehat{\mathbf{U}}_l^{1,*} \mathbf{U}_l^0 \in \mathbb{C}^{\widehat{r}_l \times r_l^0}$

else

$[\widehat{Y}_{\tau_i}^1, \widehat{C}_{\tau_i}^0] = \textit{Rank-augmenting TTN integrator}(\tau_i, Y_{\tau_i}^0, F_{\tau_i}, t_0, t_1)$;

% `recursive computation on subtrees`

compute an orthonormal basis $\widehat{\mathbf{Q}}_{\tau_i}$ of rank $\widehat{r}_{\tau_i} \le 2r_{\tau_i}^0$ of the range of $\big(\mathbf{Mat}_0(\widehat{C}_{\tau_i}^1)^\top, \mathbf{Mat}_0(\widehat{C}_{\tau_i}^0)^\top\big)$, where $\widehat{C}_{\tau_i}^1$ is the connection tensor of $\widehat{Y}_{\tau_i}^1$;

set $\widehat{\mathbf{U}}_{\tau_i}^1 = \mathbf{Mat}_0(\widehat{X}_{\tau_i}^1)^\top$, where the orthonormal TTN $\widehat{X}_{\tau_i}^1$ is obtained from $\widehat{Y}_{\tau_i}^1$ by replacing the connection tensor with $\widehat{C}_{\tau_i} = \mathit{Ten}_0(\widehat{\mathbf{Q}}_{\tau_i}^\top)$;

set $\widehat{\mathbf{M}}_{\tau_i} = \widehat{\mathbf{U}}_{\tau_i}^{1,*} \mathbf{U}_{\tau_i}^0 \in \mathbb{C}^{\widehat{r}_{\tau_i} \times r_{\tau_i}^0}$ (computed as $\langle \widehat{X}_{\tau_i}^1, X_{\tau_i}^0 \rangle$)

end

end

Algorithm 7: Subflow Ψ_τ (augment and update the connection tensor)

Data: tree $\tau = (\tau_1, \dots, \tau_m)$, connection tensor $C_\tau^0 \in \mathbb{C}^{r_\tau^0 \times r_{\tau_1}^0 \times \cdots \times r_{\tau_m}^0}$, augmented basis matrices $\widehat{\mathbf{U}}_{\tau_i}^1$ in factorized form, auxiliary matrices $\widehat{\mathbf{M}}_{\tau_i} \in \mathbb{C}^{\widehat{r}_{\tau_i} \times r_{\tau_i}^0}$, function $F_\tau(t, Y)$, t_0, t_1

Result: connection tensors $\widehat{C}_\tau^1, \widehat{C}_\tau^0 \in \mathbb{C}^{r_\tau \times \widehat{r}_{\tau_1} \times \cdots \times \widehat{r}_{\tau_m}}$

begin

set $\widehat{C}_\tau^0 = C_\tau^0 \times_{i=1}^m \widehat{\mathbf{M}}_{\tau_i}$;

solve the $r_\tau \times \widehat{r}_{\tau_1} \times \cdots \times \widehat{r}_{\tau_m}$ tensor differential equation from t_0 to t_1

$$\dot{\widehat{C}}_\tau(t) = F_\tau\big(t, \widehat{C}_\tau(t) \bigtimes_{i=1}^m \widehat{\mathbf{U}}_{\tau_i}^1\big) \bigtimes_{i=1}^m \widehat{\mathbf{U}}_{\tau_i}^{1,*}, \quad \widehat{C}_\tau(t_0) = \widehat{C}_\tau^0;$$

% `Galerkin method on the subspace generated by all` $\widehat{\mathbf{U}}_{\tau_i}^1$

set $\widehat{C}_\tau^1 = \widehat{C}_\tau(t_1)$

end

We end this section by giving the computational complexity of the proposed algorithm.

As in [CLW21], we count the required arithmetical operations and the memory requirements. We arrive at the following result, originally coming from [CLS23, Lemma 4.1]. Note that we make the following assumption on the function F: For every tree tensor network $X_{\bar{\tau}}$, the function value $Z_{\bar{\tau}} = F(t, X_{\bar{\tau}})$ is approximated by a tree tensor network with ranks $s_\tau \leq cr$ with $r = \max_\tau r_\tau$ for all subtrees $\tau \leq \bar{\tau}$ with a moderate constant c.

Lemma 4.2: Computational complexity

Let d be the order of the tensor $A(t)$ (i.e., the number of leaves of the tree $\bar{\tau}$), $l < d$ the number of levels (i.e., the height of the tree $\bar{\tau}$), and let $n = \max_\ell n_\ell$ be the maximal dimension, $r = \max_\tau r_\tau$ the maximal rank and m the maximal order of the connection tensors. Under the above assumption on the approximation of F, one time step of the rank-adaptive tree tensor integrator given by algorithms 5–7 and algorithm 1 requires

- $O(dr(n + r^m))$ storage,
- $O(ld)$ tensorizations/matricizations of matrices/tensors with $\leq r^{m+1}$ entries,
- $O(ld^2r^2(n + r^m))$ arithmetical operations and
- $O(d)$ evaluations of the function F,

provided the differential equations in algorithms 6 and 7 are solved approximately using a fixed number of function evaluations per time step.

4.4 Fixed-rank BUG integrator for Tree Tensor Networks

The fixed-rank BUG (unconventional) integrator for matrices and Tucker tensors from [CL21] can be extended to tree tensor networks in the same way as above. This yields a fixed-rank TTN integrator which differs from the rank-adaptive algorithm only in that the matrices of doubled dimension $\left(Y_l(t_1)^\top, \mathbf{U}_l^0\right)$ and $\left(\mathbf{Mat}_0(\widehat{C}_{\tau_i}^1)^\top, \mathbf{Mat}_0(\widehat{C}_{\tau_i}^0)^\top\right)$ in algorithm 6, which contain new and old basis, are replaced by taking only the new basis $Y_l(t_1)^\top$ and $\mathbf{Mat}_0(\widehat{C}_{\tau_i}^1)^\top$. The truncation function Θ is then not needed. The subflow Ψ_τ is the same as in algorithm 7, while the subflow $\Phi_\tau^{(i)}$ changes and has the form

Algorithm 8: Subflow $\Phi_\tau^{(i)}$ fixed-rank BUG integrator for tree tensor networks

Data: tree $\tau = (\tau_1, \cdots, \tau_m)$, TTN $Y_\tau^0 = C_\tau^0 \times_0 \mathbf{I} \times_{i=1}^d \mathbf{U}_{\tau_i}^0$ with $\mathbf{U}_{\tau_i}^0 = \mathbf{Mat}_0(X_{\tau_i}^0)^\top$, function $F_\tau(t,\cdot)$, t_0, t_1

Result: TTN $Y_\tau^1 = C_\tau^1 \times_0 \mathbf{I} \times_{i=1}^d \mathbf{U}_{\tau_i}^1$ with $\mathbf{U}_{\tau_i}^1 = \mathbf{Mat}_0(X_{\tau_i}^1)^\top$, $r_{\tau_i} \times r_{\tau_i}$ matrix $\mathbf{M}_{\tau_i}$

begin

compute a QR-decomposition $\mathbf{Mat}_i(C_\tau^0)^\top = \mathbf{Q}_{\tau_i}^0 \mathbf{S}_{\tau_i}^{0,\top}$

set $Y_{\tau_i}^0 = X_{\tau_i}^0 \times_0 \mathbf{S}_{\tau_i}^{0,\top}$

if $\tau_i = l$ *is a leaf* **then**

solve the $n_l \times r_l$ ODE

$$\dot{Y}_{\tau_i}(t) = F_{\tau_i}(t, Y_{\tau_i}(t)), \quad Y_{\tau_i}(t_0) = Y_{\tau_i}^0$$

compute a QR-decomposition $Y_{\tau_i}(t_1) = \mathbf{Q}_{\tau_i}^1 \mathbf{R}_{\tau_i}^1$

set $\mathbf{U}_{\tau_i}^1 = \mathbf{Q}_{\tau_i}^1$

set $\mathbf{M}_{\tau_i} = \mathbf{U}_{\tau_i}^{1,*} \mathbf{U}_{\tau_i}^0$

else

$Y_{\tau_i}^1 =$ *fixed-rank BUG integrator for* $TTN(\tau_i, Y_{\tau_i}^0, F_{\tau_i}, t_0, t_1)$

compute a QR-decomposition $\mathbf{Mat}_0(C_{\tau_i}^1)^\top = \mathbf{Q}_{\tau_i}^1 \mathbf{R}_{\tau_i}^1$, where $C_{\tau_i}^1$ is the root tensor of $Y_{\tau_i}^1$

set $Ten_0(\mathbf{Q}_{\tau_i}^{1,\top})$ as the root tensor of $Y_{\tau_i}^1$

set $\mathbf{U}_{\tau_i}^1 = Y_{\tau_i}^1$

set $\mathbf{M}_{\tau_i} = \mathbf{U}_{\tau_i}^{1,*} \mathbf{U}_{\tau_i}^0$

end

set $C_\tau^1 = C_\tau^0$

end

Note that in algorithm 8, the basis matrices are denoted by $\mathbf{U}_\tau^1$ without a hat, since they are not augmented. The core update in the corresponding subflow Ψ_τ is therefore performed using the basis matrices $\mathbf{U}_{\tau_i}^1$, whereas for the rank-adaptive BUG the basis matrices $\widehat{\mathbf{U}}_{\tau_i}^1$ are used. In contrast to the rank-adaptive integrator, the unconventional integrator keeps the ranks fixed during the propagation.

4.5 Constructing F_{τ_i} and $Y_{\tau_i}^0$

For each $\tau = (\tau_1, \ldots, \tau_m) \leq \bar{\tau}$, where $\bar{\tau}$ denotes a tree with d leaves, we define the tensor space

$$\mathcal{V}_\tau := \mathbb{C}^{r_\tau \times n_{\tau_1} \times \cdots \times n_{\tau_m}},$$

where $n_\tau = \prod_{j=1}^m n_{\tau_j}$. Note that for $r_{\bar{\tau}} = 1$ the space $\mathcal{V}_{\bar{\tau}}$ is isomorphic to $\mathbb{C}^{n_1 \times \cdots \times n_d}$.
In the following we will assume that $F = F_{\bar{\tau}} : [0, T_{\text{end}}] \times \mathcal{V}_\tau \to \mathcal{V}_\tau$ is given and maps a TTN to another TTN of possibly larger rank. For a given subtree $\tau = (\tau_1, \dots, \tau_m)$ we further assume by induction that we already have constructed F_τ and the initial data Y_τ^0. For $i = 1, \dots, m$ we want to construct functions F_{τ_i} and initial data $Y_{\tau_i}^0$. F_{τ_i} can be seen as a reduced operator acting only on the subtree τ_i, while $Y_{\tau_i}^0$ is the initial data on the subtree τ_i.

This section relies mainly on [CLW21, section 4]. In [CLS23] the ideas on how to construct the functions F_{τ_i} and the initial data $Y_{\tau_i}^0$ are summarized and used but not proven. Therefore, we give a detailed insight based on [CLW21], with the difference that we here consider tree tensor networks with complex entries.

4.5.1 Constructing initial data $Y_{\tau_i}^0$

Suppose we have given the initial data $Y_\tau^0 = C_\tau^0 \times_0 \mathbf{I}_\tau \times_{j=1}^m \mathbf{U}_{\tau_j}^0$ in a tree tensor network representation. First, we compute the QR-decomposition

$$\mathbf{Mat}_i(C_\tau^0)^\top = \mathbf{Q}_{\tau_i}^0 \mathbf{S}_{\tau_i}^{0,\top},$$

where $\mathbf{S}_{\tau_i}^{0,\top} \in \mathbb{C}^{r_{\tau_i} \times r_{\tau_i}}$ and $\mathbf{Q}_{\tau_i}^0 \in \mathbb{C}^{r_{\tau_i} \times r_{\neg\tau_i}}$ with $r_{\neg\tau_i} = \prod_{j \neq i} r_{\tau_j}$. Clearly, the matrix $\mathbf{Q}_{\tau_i}^0$ has orthonormal columns. Now we compute

$$\begin{aligned} Y_\tau^0 &= C_\tau^0 \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^0 = Ten_i\left(\mathbf{S}_{\tau_i}^0 \mathbf{Q}_{\tau_i}^{0,\top}\right) \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^0 \\ &= Ten_i\left(\mathbf{Q}_{\tau_i}^{0,\top}\right) \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^0 \times_i \mathbf{S}_{\tau_i}^0 = Ten_i\left(\mathbf{Q}_{\tau_i}^{0,\top}\right) \times_0 \mathbf{I}_\tau \bigtimes_{j \neq i} \mathbf{U}_{\tau_j}^0 \times_i (\mathbf{U}_{\tau_i}^0 \mathbf{S}_{\tau_i}^0). \end{aligned}$$

Therefore we have the SVD-like decomposition

$$\mathbf{Mat}_i(Y_\tau^0) = \mathbf{U}_{\tau_i}^0 \mathbf{S}_{\tau_i}^0 \mathbf{V}_{\tau_i}^{0,*},$$

where $\mathbf{V}_{\tau_i}^0$ is the matrix

$$\mathbf{V}_{\tau_i}^0 := \mathbf{Mat}_i\left(Ten_i(\mathbf{Q}_{\tau_i}^{0,\top}) \times_0 \mathbf{I}_\tau \bigtimes_{j \neq i} \mathbf{U}_{\tau_j}^0 \right)^* \in \mathbb{C}^{r_\tau n_{\neg\tau_i} \times r_{\tau_i}} \tag{4.1}$$

and $n_{\neg\tau_i} = \prod_{j \neq i} n_{\tau_j}$.
Now recall the the differential equation for $\mathbf{K}_{\tau_i}$ from the subflow $\Phi^{(i)}$ of the Tucker

integrator (algorithm 3). We have

$$\begin{aligned}\dot{\mathbf{K}}_{\tau_i}(t) &= \boldsymbol{F}_{\tau_i}(t, \mathbf{K}_{\tau_i}(t)),\\ \mathbf{K}_{\tau_i}(t_0) &= \mathbf{U}^0_{\tau_i}\mathbf{S}^0_{\tau_i},\end{aligned} \tag{4.2}$$

where

$$\boldsymbol{F}_{\tau_i}(t, \mathbf{K}_{\tau_i}) = \mathbf{Mat}_i\left(F_\tau(t, \mathit{Ten}_i(\mathbf{K}_{\tau_i}\mathbf{V}^{0,*}_{\tau_i}))\right)\mathbf{V}^0_{\tau_i}. \tag{4.3}$$

Using $\mathbf{U}^0_{\tau_i} = \mathbf{Mat}_0(X^0_{\tau_i})^\top$ we obtain for the initial data

$$\mathbf{K}_{\tau_i}(t_0) = \mathbf{U}^0_{\tau_i}\mathbf{S}^0_{\tau_i} = \mathbf{Mat}_0(X^0_{\tau_i})^\top\mathbf{S}^0_{\tau_i} = \mathbf{Mat}_0\left(X^0_{\tau_i} \times_0 \mathbf{S}^{0,\top}_{\tau_i}\right)^\top,$$

which is the same as the initial data chosen in the subflow $\Phi^{(i)}_\tau$ in algorithm 6. Tensorizing both equations in equation (4.2) and substituting $Y_{\tau_i}(t) = \mathit{Ten}_0(\mathbf{K}_{\tau_i}(t)^\top)$ into the equation gives us the differential equation as in algorithm 6

$$\begin{aligned}\dot{Y}_{\tau_i}(t) &= F_{\tau_i}(t, Y_{\tau_i}(t))\\ Y_{\tau_i}(t_0) &= Y^0_{\tau_i} := X^0_{\tau_i} \times_0 \mathbf{S}^{0,\top}_{\tau_i},\end{aligned}$$

where by (4.3) we have

$$\begin{aligned}F_{\tau_i}(t, Y_{\tau_i}) &= \mathit{Ten}_0\left(F_{\tau_i}(t, \mathbf{Mat}_0(Y_{\tau_i})^\top)\right)\\ &= \mathit{Ten}_0\left(\left(\mathbf{Mat}_i\left(F_\tau(t, \mathit{Ten}_i(\mathbf{Mat}_0(Y_{\tau_i})^\top\mathbf{V}^{0,*}_{\tau_i}))\right)\mathbf{V}^0_{\tau_i}\right)^\top\right).\end{aligned} \tag{4.4}$$

The construction of the function F_{τ_i} becomes more clear when introducing a prolongation and restriction function from the following subsection.

4.5.2 Prolongation and restriction

We define the prolongation $\pi_{\tau,i}$ and restriction $\pi^\dagger_{\tau,i}$ by

$$\begin{aligned}\pi_{\tau,i}(Y_{\tau_i}) &:= \mathit{Ten}_i(\mathbf{Mat}_0(Y_{\tau_i})^\top\mathbf{V}^{0,*}_{\tau_i}) \in \mathcal{V}_\tau, \text{ for } Y_{\tau_i} \in \mathcal{V}_{\tau_i}\\ \pi^\dagger_{\tau,i}(Z_\tau) &:= \mathit{Ten}_0((\mathbf{Mat}_i(Z_\tau)\mathbf{V}^0_{\tau_i})^\top) \in \mathcal{V}_{\tau_i}, \text{ for } Z_\tau \in \mathcal{V}_\tau.\end{aligned}$$

By the definition of the restriction, (4.2) and the substitution $Y_{\tau_i}(t) = \mathit{Ten}_0(\mathbf{K}_{\tau_i}(t)^\top)$ from above, the initial value can be re-cast as follows

$$\begin{aligned}Y^0_{\tau_i} &= \mathit{Ten}_0\left(\mathbf{K}_{\tau_i}(t_0)^\top\right) = \mathit{Ten}_0\left((\mathbf{U}^0_{\tau_i}\mathbf{S}^0_{\tau_i})^\top\right)\\ &= \mathit{Ten}_0\left((\mathbf{Mat}_i(Y^0_\tau)\mathbf{V}^0_{\tau_i})^\top\right) = \pi^\dagger_{\tau,i}(Y^0_\tau).\end{aligned} \tag{4.5}$$

By the above definitions and the computation above, we arrive at the more compact formulation for F_{τ_i} of (4.4) and $Y^0_{\tau_i}$ by

Definition 4.1: F_{τ_i} and $Y^0_{\tau_i}$

Let $\tau = (\tau_1, \dots, \tau_m) \leq \bar{\tau}$ be a tree and suppose the function F_τ and initial data Y^0_τ is given. Further, let the prolongation and restriction be defined as above. Then we define recursively

$$F_{\tau_i} := \pi^\dagger_{\tau,i} \circ F_\tau \circ \pi_{\tau,i}, \tag{4.6}$$

$$Y^0_{\tau_i} := \pi^\dagger_{\tau,i}(Y^0_\tau) \tag{4.7}$$

for all $i = 1, \dots, m$.

Further we will recall some interesting properties of the prolongation and restriction. These can be found for the real case in [CLW21] as lemma 4.1, lemma 4.3 and lemma 4.4. Here, we consider the extension to the complex setting.

Lemma 4.3

If the initial tree tensor network $Y^0_{\bar{\tau}}$ has full tree rank $(r_\sigma)_{\sigma \leq \bar{\tau}}$, then Y^0_τ, as defined in (4.7), has full tree rank $(r_\sigma)_{\sigma \leq \tau}$ for every subtree $\tau \leq \bar{\tau}$.

Proof.
Let $\tau = (\tau_1, \dots, \tau_m) \leq \bar{\tau}$ and $i = 1, \dots, m$. By the above derivation we have

$$Y^0_{\tau_i} = X^0_{\tau_i} \times_0 S^{0,\top}_{\tau_i},$$

where $X_{\tau_i} = Ten_0(\mathbf{U}^{0,\top}_{\tau_i})$ is of full tree rank. If Y^0_τ has full tree rank, then $S^0_{\tau_i}$ is invertible and therefore $Y^0_{\tau_i}$ has full tree rank. By induction over the height of the tree, we find that for every subtree $\tau \leq \bar{\tau}$, the restricted initial tensor Y^0_τ has full tree rank. □

The latter lemma can be restated by saying that for a tree $\tau = (\tau_1, \dots, \tau_m)$ and $Y^0_\tau \in \mathcal{M}_\tau$, the restriction $\pi^\dagger_{\tau,i}(Y^0_\tau)$ is in $\mathcal{M}_{\tau_i}$. Note that this is not generally true for arbitrary inputs, see [CLW21, section 4] for details. However, for the prolongation, we have the following property.

Lemma 4.4

Let $\tau = (\tau_1, \dots, \tau_m)$. If $Y_{\tau_i} \in \mathcal{M}_{\tau_i}$, then the prolongation $\pi_{\tau,i}(Y_{\tau_i})$ is in $\mathcal{M}_\tau$, for $i = 1, \dots, m$.

Proof. Using the fact that $\mathbf{Mat}_0(Y_{\tau_i})^\top = \mathbf{U}_{\tau_i}$, the definition of $\mathbf{V}^0_{\tau_i}$ and the unfolding formula (2.4) we compute

$$\begin{aligned}\pi_{\tau_i}(Y_{\tau_i}) &= Ten_i(\mathbf{U}_{\tau_i}\mathbf{V}^{0,*}_{\tau_i}) \\ &= Ten_i\left(\mathbf{U}_{\tau_i}\,\mathbf{Mat}_i\left(Ten_i(\mathbf{Q}^{0,\top}_{\tau_i})\times_0 \mathbf{I}_\tau \bigtimes_{j\neq i}\mathbf{U}^0_{\tau_j}\right)\right) \\ &= Ten_i(\mathbf{Q}^{0,\top}_{\tau_i})\times_0 \mathbf{I}_\tau \bigtimes_{j\neq i}\mathbf{U}^0_{\tau_j}\times_i \mathbf{U}_{\tau_i}.\end{aligned}$$

As $Ten_i(\mathbf{Q}^{0,\top}_{\tau_i})$ and all $\mathbf{U}_{\tau_j}$ for $j = 1,\dots,m$ are by assumption of full tree rank, the full object is of full tree rank. Thus, $\pi_{\tau,i}(Y_{\tau_i})$ lies in $\mathcal{M}_\tau$. □

Next, we report two further properties of the restriction and prolongation which are essential tools for the analysis of the recursive algorithms.

Lemma 4.5

Let $\tau = (\tau_1,\dots,\tau_m)$ and $i = 1,\dots,m$. The restriction $\pi^\dagger_{\tau,i} : \mathcal{V}_\tau \to \mathcal{V}_{\tau_i}$ is both a left inverse and the adjoint (with respect to the tensor Euclidean inner product) of the prolongation $\pi_{\tau,i} : \mathcal{V}_{\tau_i} \to \mathcal{V}_\tau$, that is,

$$\pi^\dagger_{\tau,i}(\pi_{\tau,i}(Y_{\tau_i})) = Y_{\tau_i} \quad \text{for all } Y_{\tau_i} \in \mathcal{V}_{\tau_i} \tag{4.8}$$

$$\langle \pi_{\tau,i}(Y_{\tau_i}), Z_\tau\rangle_{\mathcal{V}_\tau} = \langle Y_{\tau_i}, \pi^\dagger_{\tau,i}(Z_\tau)\rangle_{\mathcal{V}_{\tau_i}} \quad \text{for all } Y_{\tau_i} \in \mathcal{V}_{\tau_i}, Z_\tau \in \mathcal{V}_\tau. \tag{4.9}$$

Moreover, $\|\pi_{\tau,i}(Y_{\tau_i})\|_{\mathcal{V}_\tau} = \|Y_{\tau_i}\|_{\mathcal{V}_{\tau_i}}$ and $\|\pi^\dagger_{\tau,i}(Z_\tau)\|_{\mathcal{V}_{\tau_i}} \le \|Z_\tau\|_{\mathcal{V}_\tau}$, where the norms are the tensor Euclidean norms.

Proof.
By equation (4.1) we readily obtain that $\mathbf{V}^{0,*}_{\tau_i}\mathbf{V}^0_{\tau_i} = \mathbf{I}$. Let $Y_{\tau_i} \in \mathcal{V}_{\tau_i}$, we obtain

$$\begin{aligned}\pi^\dagger_{\tau,i}(\pi_{\tau,i}(Y_{\tau_i})) &= Ten_0((\mathbf{Mat}_i(\,Ten_i(\mathbf{Mat}_0(Y_{\tau_i})^\top\mathbf{V}^{0,*}_{\tau_i}))\mathbf{V}^0_{\tau_i})^\top) \\ &= Ten_0((\mathbf{Mat}_0(Y_{\tau_i})^\top \underbrace{\mathbf{V}^{0,*}_{\tau_i}\mathbf{V}^0_{\tau_i}}_{=\mathbf{I}})^\top) = Y_{\tau_i},\end{aligned}$$

which proofs (4.8). Using that Ten_i function is the adjoint of $\mathbf{Mat}_i$ function for the Frobenius inner product and that taking transposes in both matrices does not change

the Frobenius inner product we obtain

$$\begin{aligned}\langle \pi_{\tau,i}(Y_{\tau_i}), Z_\tau\rangle_{\mathcal{V}_\tau} &= \langle Ten_i(\mathbf{Mat}_0(Y_{\tau_i})^\top \mathbf{V}_{\tau_i}^{0,*}), Z_\tau\rangle_{V_\tau} \\ &= \langle \overline{\mathbf{V}_{\tau_i}^0}\mathbf{Mat}_0(Y_{\tau_i}), \mathbf{Mat}_i(Z_\tau)^\top\rangle_{\mathcal{V}_\tau} \\ &= \langle \mathbf{Mat}_0(Y_{\tau_i}), \overline{\mathbf{V}_{\tau_i}^{0,*}}\,\mathbf{Mat}_i(Z_\tau)^\top\rangle_{\mathcal{V}_{\tau_i}} \\ &= \langle Y_{\tau_i}, Ten_0((\mathbf{Mat}_i(Z_\tau)\mathbf{V}_{\tau_i}^0)^\top)\rangle_{\mathcal{V}_{\tau_i}} = \langle Y_{\tau_i}, \pi_{\tau,i}^\dagger(Z_\tau)\rangle_{\mathcal{V}_{\tau_i}},\end{aligned}$$

which proofs (4.9).
With the same arguments as before and noting that $\mathbf{V}_{\tau_i}^0$ has orthonormal columns, we compute

$$\begin{aligned}\|\pi_{\tau,i}(Y_{\tau_i})\|_{\mathcal{V}_\tau} &= \left\| Ten_i(\mathbf{Mat}_0(Y_{\tau_i})^\top \mathbf{V}_{\tau_i}^{0,*})\right\|_{V_\tau} \\ &= \left\|\mathbf{Mat}_0(Y_{\tau_i})^\top \mathbf{V}_{\tau_i}^{0,*}\right\|_{\mathcal{V}_\tau} = \|\mathbf{Mat}_0(Y_{\tau_i})\|_{\mathcal{V}_{\tau_i}} = \|Y_{\tau_i}\|_{\mathcal{V}_{\tau_i}}.\end{aligned}$$

Last, we note that $\left\|\mathbf{V}_{\tau_i}^{0,*}\right\|_2 = 1$ and that generally it holds $\|\mathbf{AB}\| \le \|\mathbf{A}\|_2 \|\mathbf{B}\|$ to obtain

$$\begin{aligned}\|\pi_{\tau,i}^\dagger(Z_\tau)\|_{\mathcal{V}_{\tau_i}} &= \left\| Ten_0(\mathbf{Mat}_0(Y_{\tau_i})^\top \mathbf{V}_{\tau_i}^{0,*})\right\|_{V_{\tau_i}} = \left\|\mathbf{Mat}_0(Y_{\tau_i})^\top \mathbf{V}_{\tau_i}^{0,*}\right\|_{V_{\tau_i}} \\ &\le \left\|\mathbf{V}_{\tau_i}^{0,*}\right\|_2 \left\|\mathbf{Mat}_i(Z_\tau)^\top\right\|_{\mathcal{V}_\tau} = \|Z_\tau\|_{\mathcal{V}_\tau}.\end{aligned}$$

□

As $\pi_{\tau,i}^\dagger$ is the adjoint of $\pi_{\tau,i}$ by lemma 4.5, we have that if F_τ is selfadjoint then F_{τ_i} remains selfadjoint. Therefore if we consider a tensor Schrödinger equation, i.e. $F_{\bar\tau} = H$ is a Hamiltonian, all F_τ for $\tau \le \bar\tau$ are again a Hamiltonian of smaller size. The sub-Hamiltonians F_τ only act on the particles associated with the subtree τ and keep the remaining part of the tree $\bar\tau$ fixed.

4.5.3 Efficient computation of prolongation and restriction

Prolongation

Recall the definition for the matrix $\mathbf{V}_{\tau_i}^0$ from (4.1)

$$\mathbf{V}_{\tau_i}^0 = \mathbf{Mat}_i\left(Ten_i(\mathbf{Q}_{\tau_i}^{0,\top}) \times_0 \mathbf{I}_\tau \bigtimes_{j\neq i} \mathbf{U}_{\tau_j}^0\right)^* \in \mathbb{C}^{r_\tau n_{\neg\tau_i} \times r_{\tau_i}}.$$

Using the definition of the prolongation and the unfolding formula (2.4) we have

$$\begin{aligned}
\pi_{\tau,i}(Y) &= Ten_i(\mathbf{Mat}_0(Y)^\top \mathbf{V}_{\tau_i}^{0,*}) \\
&= Ten_i\left(\mathbf{Mat}_0(Y)^\top \mathbf{Mat}_i\left(Ten_i(\mathbf{Q}_{\tau_i}^{0,\top}) \times_0 \mathbf{I}_\tau \bigtimes_{j\neq i} \mathbf{U}_{\tau_j}^0\right)\right) \\
&= Ten_i\left(\mathbf{Mat}_i\left(Ten_i(\mathbf{Q}_{\tau_i}^{0,\top}) \times_i \mathbf{Mat}_0(Y)^\top \bigtimes_{j\neq i} \mathbf{U}_{\tau_j}^0\right)\right) \\
&= Ten_i(\mathbf{Q}_{\tau_i}^{0,\top}) \times_i \mathbf{Mat}_0(Y)^\top \bigtimes_{j\neq i} \mathbf{U}_{\tau_j}^0.
\end{aligned}$$

This computation shows that the prolongation on the tree tensor network $Y \in \mathcal{V}_{\tau_i}$ yields a larger tree tensor network in $\mathcal{V}_\tau$ with the core tensor $\mathbf{Q}_{\tau_i}^{0,\top}$. This allows computing the prolongation without constructing the possibly inaccessible matrix $\mathbf{V}_{\tau_i}^{0,*}$.

Restriction

Suppose we have a tree tensor network $Z_\tau = D_\tau \times_0 \mathbf{I}_\tau \times_{i=1}^m \mathbf{W}_{\tau_i}$. By inserting the definition for $\mathbf{V}_{\tau_i}^0$ and using again the unfolding formula (2.4) we obtain

$$\begin{aligned}
\pi_{\tau,i}^\dagger(Z_\tau) &= Ten_0((\mathbf{Mat}_i(Z_\tau)\mathbf{V}_{\tau_i}^0)^\top) = Ten_0(\mathbf{V}_{\tau_i}^{0,\top}\, \mathbf{Mat}_i(Z_\tau)^\top) \\
&= Ten_0\left(\overline{\mathbf{Mat}_i(\, Ten_i(\mathbf{Q}_{\tau_i}^{0,\top}) \times_0 \mathbf{I}_\tau \bigtimes_{j\neq i} \mathbf{U}_{\tau_j}^0)}\, \mathbf{Mat}_i(Z_\tau)^\top\right) \\
&= Ten_0\left(\mathbf{Q}_{\tau_i}^{0,*}(\bigotimes_{j\neq i} \mathbf{U}_{\tau_i}^{0,*} \otimes \mathbf{I}_\tau)\, \mathbf{Mat}_i(Z_\tau)^\top\right).
\end{aligned}$$

Inserting now the Z_τ and applying the unfolding formula (2.4) again we get

$$\begin{aligned}
\pi_{\tau,i}^\dagger(Z_\tau) &= Ten_0\left(\mathbf{Q}_{\tau_i}^{0,*}(\bigotimes_{j\neq i} \mathbf{U}_{\tau_j}^{0,*} \otimes \mathbf{I}_\tau)(\bigotimes_{j\neq i} \mathbf{W}_{\tau_j} \otimes \mathbf{I}_\tau)\, \mathbf{Mat}_i(D_\tau)^\top \mathbf{W}_{\tau_i}\right) \\
&= Ten_0\left(\mathbf{Q}_{\tau_i}^{0,*}(\bigotimes_{j\neq i} \mathbf{U}_{\tau_j}^{0,*}\mathbf{W}_{\tau_j} \otimes \mathbf{I}_\tau)\, \mathbf{Mat}_i(D_\tau)^\top \mathbf{W}_{\tau_i}\right) \\
&= Ten\left(\mathbf{Q}_{\tau_i}^{0,*}\, \mathbf{Mat}_i\left(D_\tau \times_0 \mathbf{I}_\tau \bigtimes_{j\neq i} (\mathbf{U}_{\tau_j}^{0,*}\mathbf{W}_{\tau_j})^\top\right)^\top \mathbf{W}_{\tau_i}^\top\right).
\end{aligned}$$

We define the matrix

$$\mathbf{R}_{\tau_i} := \mathbf{Q}_{\tau_i}^{0,*}\, \mathbf{Mat}_i(D_\tau \times_0 \mathbf{I}_\tau \bigtimes_{j\neq i} (\mathbf{U}_{\tau_j}^{0,*}\mathbf{W}_{\tau_j})^\top)^\top, \tag{4.10}$$

where the matrix products $\mathbf{U}_{\tau_j}^{0,*}\mathbf{W}_{\tau_j}$ can be computed as an inner product as described in subsection 2.4.3. By definition of a tree tensor network, there exists a TTN Z_{τ_i} such

that $\mathbf{W}_{\tau_i} = \mathbf{Mat}_0(Z_{\tau_i})^\top$. Inserting this we obtain for the restriction

$$\begin{aligned}\pi^\dagger_{\tau,i}(Z_\tau) &= Ten_0(\mathbf{R}_{\tau_i}\mathbf{W}^\top_{\tau_i}) = Ten_0(\mathbf{R}_{\tau_i}\,\mathbf{Mat}_0(Z_{\tau_i}))\\ &= Ten_0(\mathbf{Mat}_0(Z_{\tau_i}\times_0\mathbf{R}_{\tau_i})) = Z_{\tau_i}\times_0\mathbf{R}_{\tau_i}.\end{aligned}$$

By this we see that $\pi^\dagger_{\tau,i}(Z_\tau)$ can be efficiently computed by multiplying the TTN Z_{τ_i} in the 0-dimension with the matrix $\mathbf{R}_{\tau_i}$. If D_{τ_i} is the core tensor of Z_{τ_i}, the result is equivalent to replace the root tensor of Z_{τ_i} by $D_{\tau_i}\times_0\mathbf{R}_{\tau_i}$.

4.6 Preparatory lemma

For the analysis of the rank-adaptive tree tensor network integrator we give an essential result about the original and the augmented initial tensors, which is actually used in the algorithm. I.e. we will show that

$$Y^0_\tau = C^0_\tau\times_0\mathbf{I}_\tau\bigtimes_{i=1}^m\mathbf{U}^0_{\tau_i}$$

equals the tree tensor network with augmented connecting tensor $\widehat{C}^0_\tau = C^0_\tau\times_{i=1}^m\widehat{\mathbf{U}}^{1,*}_{\tau_i}\mathbf{U}^0_{\tau_i}$ from algorithm 7 and basis matrices $\widehat{\mathbf{U}}^1_{\tau_i}$ from algorithm 6

$$\widehat{Y}^0_\tau = \widehat{C}^0_\tau\times_0\mathbf{I}_\tau\bigtimes_{i=1}^m\widehat{\mathbf{U}}^1_{\tau_i}.$$

The following results and its proof can be found as lemma 6.1 in [CLS23].

Lemma 4.6

For the initial tree tensor network Y^0_τ it holds

$$\widehat{Y}^0_\tau = Y^0_\tau.$$

Proof.

To prove the statement, we first show over an induction over the height of the tree that

$$\mathrm{Range}(\mathbf{U}^0_{\tau_i})\subseteq\mathrm{Range}(\widehat{\mathbf{U}}^1_{\tau_i}). \tag{4.11}$$

We start the induction with a tree of height zero, i.e. $\tau = l$ a leaf. By the definition of $\widehat{\mathbf{U}}^1_l$ from algorithm 6 it is obvious that (4.11) holds true. Now, for $\tau_i = \sigma = (\sigma_1,\dots,\sigma_k)$ we use the induction hypothesis that $\mathrm{Range}(\mathbf{U}^0_{\sigma_j})\subseteq\mathrm{Range}(\widehat{\mathbf{U}}^1_{\sigma_j})$, for all $j = 1,\dots,k$.

This can be written as

$$\mathbf{U}_{\sigma_j}^0 = \widehat{\mathbf{U}}_{\sigma_j}^1 \widehat{\mathbf{U}}_{\sigma_j}^{1,*} \mathbf{U}_{\sigma_j}^0 = \widehat{\mathbf{U}}_{\sigma_j}^1 \widehat{\mathbf{M}}_{\sigma_j}.$$

By definition of a tree tensor network and applying (2.4), we have for $Y_\sigma = C_\sigma^0 \times_0 \mathbf{I}_\sigma \times_{j=1}^k \mathbf{U}_{\sigma_j}^0$ and $\mathbf{U}_\sigma^0 = \mathbf{Mat}_0(Y_\sigma^0)^\top$ that

$$\begin{aligned}\mathbf{U}_\sigma^0 &= \bigotimes_{j=1}^k \mathbf{U}_{\sigma_j}^0 \, \mathbf{Mat}_0(C_\sigma^0)^\top = \bigotimes_{j=1}^k \widehat{\mathbf{U}}_{\sigma_j}^1 \widehat{\mathbf{M}}_{\sigma_k} \, \mathbf{Mat}_0(C_\sigma^0)^\top \\ &= \bigotimes_{j=1}^k \widehat{\mathbf{U}}_{\sigma_j}^1 \, \mathbf{Mat}_0(C_\sigma^0 \times_{j=1}^k \widehat{\mathbf{M}}_{\sigma_k})^\top = \bigotimes_{j=1}^k \widehat{\mathbf{U}}_{\sigma_j}^1 \, \mathbf{Mat}_0(\widehat{C}_\sigma^0)^\top,\end{aligned}$$

where $\widehat{C}_\sigma^0$ is the initial data in algorithm 7. On the other hand, in algorithm 6 we construct

$$\widehat{\mathbf{U}}_\sigma^1 = \bigotimes_{j=1}^k \widehat{\mathbf{U}}_{\sigma_j}^1 \, \mathbf{Mat}_0(\widehat{C}_\sigma)^\top,$$

where the columns of $\mathbf{Mat}_0(\widehat{C}_\sigma)^\top$ form an orthogonal basis of the range of the matrix $(\mathbf{Mat}_0(\widehat{C}_\sigma^1)^\top, (\mathbf{Mat}_0(\widehat{C}_\sigma^0)^\top)$. Hence, we see that

$$\mathrm{Range}(\mathbf{Mat}_0(\widehat{C}_\sigma^0)^\top) \subseteq \mathrm{Range}(\mathbf{Mat}_0(\widehat{C}_\sigma)^\top).$$

Having these representations for $\mathbf{U}_\sigma^0$ and $\widehat{\mathbf{U}}_\sigma^1$, this implies formula 4.11.
We conclude the proof by observing that

$$\begin{aligned}\widehat{Y}_\tau^0 &= \left(C_\tau \times_{i=1}^m \widehat{\mathbf{U}}_{\tau_i}^{1,*} \mathbf{U}_{\tau_i}^0 \right) \times_0 \mathbf{I}_\tau \times_{i=1}^m \widehat{\mathbf{U}}_{\tau_i}^1 \\ &= C_\tau \times_0 \mathbf{I}_\tau \times_{i=1}^m \widehat{\mathbf{U}}_{\tau_i}^1 \widehat{\mathbf{U}}_{\tau_i}^{1,*} \mathbf{U}_{\tau_i}^0 = C_\tau \times_0 \mathbf{I}_\tau \times_{i=1}^m \mathbf{U}_{\tau_i}^0 = Y_\tau^0,\end{aligned}$$

since $\widehat{C}_\tau^0 = C_\tau^0 \times_{i=1}^m \widehat{\mathbf{M}}_{\tau_i}$ and $\mathbf{M}_{\tau_i} = \widehat{\mathbf{U}}_{\tau_i}^{1,*} \mathbf{U}_{\tau_i}^0$. □

4.7 Properties of the rank-adaptive BUG integrator for Tree Tensor Networks

In this section we will prove several properties of the rank-adaptive BUG integrator for tree tensor networks. We will show that it fulfils an exactness property and a robust error bound, which is independent of the singular values of matricizations of core tensors of the TTN. Further, we will show remarkable conservation properties. It preserves norm

and energy for Schrödinger equations up to the truncation tolerance and it diminishes the energy for gradient systems.

4.7.1 Exactness property

We will prove that the rank-adaptive TTN integrator used with tree ranks $(r_\tau)_{\tau\leq\bar{\tau}}$ from above reproduces a time-dependent family of tree tensor networks $(A(t))_{t\geq t_0}$ with same tree ranks exactly, if applied to $F(t,Y) = \dot{A}(t)$ with initial data $Y^0 = A(t_0)$. The proof goes over an induction over the height of the tree. For matrices and Tucker tensors, which are both trees of height one, such a result is already known for the rank-adaptive integrator [CKL22].

To prove the statement we need to assume a non-degeneracy condition. For a family of tree tensor networks $(A(t))_{t\geq t_0}$ of full tree rank $(r_\tau)_{\tau\leq\bar{\tau}}$ we set $Y^0_{\bar{\tau}} = A(t_0)$. The restricted tree tensor networks $A_\tau(t) := (A(t))_\tau$ are defined via the restrictions from subsection 4.5.2 associated with $Y^0_{\bar{\tau}}$ for all subtrees $\tau \leq \bar{\tau}$. By lemma 4.3 we then know that for every subtree $\tau \leq \bar{\tau}$ it holds

$$A_\tau(t_0) \text{ has full tree rank } (r_\sigma)_{\sigma\leq\tau} \text{ for every subtree } \tau \leq \bar{\tau}. \tag{4.12}$$

As the non-degeneracy condition it is now assumed that this property at time t_0 is also fulfilled at time t_1, i.e.

$$A_\tau(t_1) \text{ has full tree rank } (r_\sigma)_{\sigma\leq\tau} \text{ for every subtree } \tau \leq \bar{\tau}. \tag{4.13}$$

The following result for the rank-adaptive TTN integrator can be found in [CLS23, Section 5.2] and follows the ideas from [CLW21, Theorem 5.1].

Theorem 4.1: Exactness property

Let $A(t)$ be a continuously differentiable time-dependent family of tree tensor networks of full tree rank $(r_\tau)_{\tau\leq\bar{\tau}}$ for all $t_0 \leq t \leq t_1$ and suppose that the non-degeneracy condition (4.13) is satisfied. Then the rank-adaptive TTN integrator used with the same tree rank $(r_\tau)_{\tau\leq\bar{\tau}}$ for $F(t,Y) = \dot{A}(t)$ and without truncation is exact: starting from $Y^0 = A(t_0)$ we obtain $Y^1 = A(t_1)$.

Proof. First, we note that as the restriction $\pi^\dagger_\tau$ does not depend on time, we get

$$\dot{A}_\tau(t) = \frac{d}{dt}A_\tau(t) = (\dot{A}(t))_\tau,$$

as the derivation commutes with the (linear) restrictions. The proof goes now over an induction over the height of the tree. First, consider a tree $\tau = (\tau_1, \dots, \tau_m)$ of height one, hence the tree tensor network $A_\tau(t)$ is a Tucker tensor. By the non-degeneracy conditions (4.12) and (4.13) we know that $A_\tau(t)$ is of full multilinear rank $(r_1, \dots, r_m)$ at $t = t_0$ and $t = t_1$. The rank-adaptive TTN integrator with $F_\tau(t, Y_\tau) = \dot{A}_\tau(t)$ applied to a Tucker tensor is equivalent to the Tucker integrator in [CKL22] and hence reproduces $A_\tau(t_1)$ exactly by [CKL22, Section 5.2].
Now consider a tree of height $k \geq 2$. We use as an induction hypothesis that the rank-adaptive TTN integrator with $F_\tau(t, Y_\tau) = \dot{A}_\tau(t)$ is exact for all subtrees $\tau < \bar{\tau}$ of height strictly smaller than k. Hence, for a tree $\tau = (\tau_1, \dots, \tau_m)$ of height k all subtrees τ_i are solved exactly by the rank-adaptive TTN integrator. This reduces the integrator to the Tucker case for $F_\tau(t, Y_\tau) = \dot{A}_\tau(t)$. By (4.12) and (4.13), the tensor $A_\tau(t)$ viewed as a Tucker tensor $A_\tau(t) = C_\tau(t) \times_{i=1}^{m} \mathbf{U}_{\tau_i}(t)$, is of full multilinear rank $(r_\tau, r_{\tau_1}, \dots, r_{\tau_m})$ at times $t = t_0$ and $t = t_1$. Again by the exactness argument [CKL22, Section 5.2] follows that the rank-adaptive TTN integrator reproduces $A_\tau(t_1)$ exactly. This proves the induction.
The argument is repeated until one arrives at the maximal tree $\bar{\tau}$, which concludes the proof. □

For completeness, we note that such an exactness property is also known for the projector spitting integrator for matrices [LO14], for Tucker tensors [LVW18], tree tensor networks [CLW21] and tensor trains [LOV15]. The property also holds for the fixed-rank BUG integrator (unconventional integrator) for matrices and Tucker tensors [CL21].

Remark 4.1. The same exactness property also holds for the tree tensor network version of the fixed-rank BUG integrator (unconventional integrator) for tree tensor networks, which was discussed in section 4.4. This is proven in the same way as for the rank-adaptive BUG integrator and using the exactness property for the fixed-rank BUG integrator for Tucker tensors proven in [CL21].

4.7.2 Robust error bound

In this subsection we will extend the proof for a robust error bound for the rank-adaptive BUG integrator for Tucker tensors from [CKL22] to tree tensor networks.
For a tree $\tau = (\tau_1, \dots, \tau_m)$ we set $\mathcal{V}_\tau = \mathbb{C}^{r_\tau \times n_{\tau_1} \times \cdots \times n_{\tau_m}}$ for the tensor space and $\mathcal{M}_\tau^k = \mathcal{M}((n_\tau)_{\tau \leq \bar{\tau}}, (r_\tau^k)_{\tau \leq \bar{\tau}})$ for the manifold of tree tensor networks at the kth time step. The manifold $\mathcal{M}_\tau^k$ has the additional index k due to the changing ranks in each time step. Finally, we set $\mathcal{M}^k = \mathcal{M}_{\bar{\tau}}^k$ and $\mathcal{V} = \mathcal{V}_{\bar{\tau}}$ for the full tree $\bar{\tau}$. During the proof we make the following three assumptions; cf. [CLW21, CLS23].

1. $F : [0, t^*] \times \mathcal{V} \to \mathcal{V}$ is Lipschitz continuous and bounded, i.e.

$$\begin{aligned} \left\| F(t,Y) - F(t,\widetilde{Y}) \right\| &\leq L \left\| Y - \widetilde{Y} \right\| && \forall Y, \widetilde{Y} \in \mathcal{V} \\ \|F(t,Y)\| &\leq B && \forall Y \in \mathcal{V}. \end{aligned}$$

2. For Y near the exact solution $A(t)$ and P_Y being the orthogonal projection onto the tangent space $\mathcal{T}_Y \mathcal{M}^k$ we assume for all $t \in [t_k, t_{k+1}]$ the existence of a small $\epsilon > 0$ such that

$$\|F(t,Y) - P_Y F(t,Y)\| \leq \epsilon,$$

   in some ball $\|Y\| \leq \rho$, where it is assumed that the exact solution $A(t), 0 \leq t \leq t^*$, has a bound that is strictly smaller than ρ.

3. The error at the initial condition is bounded by

$$\left\| Y^0 - A(0) \right\| \leq \delta,$$

   for $Y^0 \in \mathcal{M}^0$.

The norm $\|\cdot\|$ used is the tensor Euclidean norm. Note that condition 1. can be weakened to a local Lipschitz condition and a local bound in a neighborhood of the exact solution $A(t)$ of the tensor differential equation (2.1) with initial data $A(t_0) = A^0 \in \mathcal{V}$.

The following result for the rank-adaptive TTN integrator can be found in [CLS23, Section 5.2] and follows the ideas from [CLW21, Theorem 6.1].

Theorem 4.2: Robust error bound for rank-adaptive BUG

Under the assumptions 1.-3., the error of the numerical approximation $Y^k \in \mathcal{M}^k$ at time $t_k = t_0 + kh$, obtained with k time steps of the rank-adaptive BUG integrator for tree tensor networks with step size $h > 0$ and rank-truncation tolerance ϑ, is bounded by

$$\|Y^k - A(t_k)\| \leq c_0\delta + c_1\epsilon + c_2 h + c_3 \vartheta/h, \quad \text{for } t_k \leq t^*.$$

All constants depend only on L, B, t^* and the tree $\bar{\tau}$ and are therefore independent of the singular values of matricizations of core tensors. This holds true provided that δ, ϵ, h and ϑ/h are so small that the above error bound guarantees that $\|Y^k\| \leq \rho$.

Proof. To prove the statement, we need that the assumptions 1.-3. are also satisfied for the reduced functions F_τ for all $\tau \leq \bar{\tau}$. This is indeed true and proven in [CLW21, Lemma 6.2].
We assume $Y^0 = A(t_0) \in \mathcal{M}$, since the difference of exact solutions of the differential equation (2.1) with initial data varying at most by δ, is bounded by $c_0\delta$ for all times $t_0 \leq t \leq t^*$, because we assumed a Lipschitz condition on F. I.e. for $Y(t)$ and $\widetilde{Y}(t)$ two exact solutions to (2.1) with initial data Y^0 and $\widetilde{Y}^0$ such that $\|Y^0 - \widetilde{Y}^0\| \leq \delta$, we have

$$\begin{aligned}
\|Y(t) - \widetilde{Y}(t)\| &= \|Y^0 - \widetilde{Y}^0 + \int_{t_0}^{t} F(s, Y(s)) - F(s, \widetilde{Y}(s))\, ds\| \\
&\leq \|Y^0 - \widetilde{Y}^0\| + \int_{t_0}^{t} \|F(s, Y(s)) - F(s, \widetilde{Y}(s))\| ds \\
&\leq \delta + L \int_{t_0}^{t} \|Y(s) - \widetilde{Y}(s)\| ds \\
&\leq \delta + L(t - t_0) e^{L(t-t_0)} \|Y(0) - \widetilde{Y}(0)\| \\
&\leq (1 + L(t^* - t_0) e^{L(t^*-t_0)})\delta.
\end{aligned}$$

Further, it suffices to show that the local error after one time step is of the order $\mathcal{O}(h(\epsilon + h))$. The final statement follows then by a standard Lady Windermere's fan argument, as in [HNW93, Section I.7 and II.3] or [KLW16].
Similar to the exactness property, the proof goes over an induction over the height of the tree. For a tree of height one, which corresponds to a Tucker tensor, the rank-adaptive BUG for tree tensor networks is equivalent to the rank-adaptive Tucker integrator from [CKL22], where the error bound is already proven.
For a tree $\tau = (\tau_1, \ldots, \tau_m)$ of height $k \geq 2$, we observe that the differential equations for Y_{τ_i} are solved approximately by intermediate tree tensor networks with lower height, for which the $\mathcal{O}(h(\epsilon + h))$ error bound holds by induction hypothesis. If the differential equations for Y_{τ_i} were solved exactly, the analysis reduces again to the Tucker integrator and we again get the $\mathcal{O}(h(\epsilon + h))$ error bound. We now have to study the influence of the inexact solution of the differential equations for Y_{τ_i}. Here we follow the same strategy as in [KLW16, Subsection 2.6.3], to see that the local error is still of magnitude $\mathcal{O}(h(\epsilon + h))$. We omit a detailed calculation since the calculation is tedious and only requires multiple applications of the triangle inequality. □

Remark 4.2. The same robust error bound, just without the dependence on ϑ, also holds for the tree tensor network version of the fixed-rank BUG integrator (unconventional integrator) for tree tensor networks, which was discussed in section 4.4. This is proven in the same way as for the rank-adaptive TTN integrator and using the robust error bound for Tucker tensors proven in [CL21] and using the simplification that the manifold stays fixed over the timesteps. We give the statement but omit the proof as it works completely analogous to the previous one.

Theorem 4.3: Robust error bound fixed rank BUG

Under the assumptions 1.-3., the error of the numerical approximation $Y^k \in \mathcal{M}$ at time $t_k = t_0 + kh$, obtained with k time steps of the fixed-rank BUG integrator for tree tensor networks with step size $h > 0$ and rank-truncation tolerance ϑ, is bounded by

$$\|Y^k - A(t_k)\| \leq c_0\delta + c_1\epsilon + c_2 h, \quad \text{for } t_k \leq t^*.$$

All constants depend only on L, B, t^* and the tree $\bar{\tau}$ and are therefore independent of the singular values of matricizations of core tensors. This holds true provided that δ, ϵ and h are so small that the above error bound guarantees that $\|Y^k\| \leq \rho$.

4.7.3 Norm preservation

If the function F on the right-hand side of the tensor differential equation (2.1) satisfies

$$\mathrm{Re}\langle Y, F(t,Y)\rangle = 0 \quad \text{for all } Y \in \mathbb{C}^{n_1\times\cdots\times n_d} \text{ and for all } t, \tag{4.14}$$

where $\langle\cdot,\cdot\rangle$ denoting the Euclidean inner product of vectorizations, then the Euclidean norm of every solution $A(t)$ of (2.1) is conserved: $\|A(t)\| = \|A(t_0)\|$ for all t. Norm conservation also holds for the rank-adaptive TTN integrator before truncation, as shown in the following result. The result was originally proven in [CLS23, Theorem 6.2].

Theorem 4.4: Norm preservation

If F satisfies (4.14), then a step of the rank-augmented TTN integrator preserves the norm: for every stepsize h and for every subtree τ of $\bar{\tau}$,

$$\|\widehat{Y}_\tau^1\| = \|Y_\tau^0\|.$$

This implies the conservation of norm if no truncation was used in the time step of the rank-adaptive BUG for TTNs. Including the truncation, by theorem 2.4, we obtain then a near-conservation of the norm up to a multiple of the truncation tolerance ϑ, i.e.

$$\|Y_\tau^0\| - c_\tau\vartheta \leq \|Y_\tau^1\| \leq \|Y_\tau^0\|,$$

with $c_\tau = \|C_\tau\|(d_\tau - 1) + 1$, as in theorem 2.4.

Proof. For every subtree $\tau = (\tau_1, \ldots, \tau_m) \leq \bar{\tau}$, recall that the reduced nonlinear operator F_τ (c.f. section 4.5) is defined by a recursion from the root to the leaves, starting from

$F_{\bar{\tau}} = F$. First, we note that

$$\operatorname{Re}\langle Y_\tau, F_\tau(t, Y_\tau)\rangle = 0 \quad \text{for all } Y_\tau \in \mathcal{V}_\tau \text{ and for all } t, \tag{4.15}$$

which follows by induction from the root to the leaves, noting

$$\langle Y_{\tau_i}, F_{\tau_i}(t, Y_{\tau_i})\rangle = \langle Y_{\tau_i}, \pi^\dagger_{\tau,i} F_\tau(t, \pi_{\tau,i} Y_{\tau_i})\rangle = \langle \pi_{\tau,i} Y_{\tau_i}, F_\tau(t, \pi_{\tau,i} Y_{\tau_i})\rangle$$

and $\pi_{\tau,i} Y_{\tau_i} \in \mathcal{V}_\tau$.

We now turn to the update of the core tensors, see algorithm 7. We have, for $k = 0$ and $k = 1$,

$$\widehat{Y}^k_\tau = \widehat{C}^k_\tau \bigtimes_{i=1}^m \widehat{\mathbf{U}}^1_{\tau_i} \quad \text{and hence} \quad \|\widehat{Y}^k_\tau\| = \|\widehat{C}^k_\tau\|,$$

since $\widehat{\mathbf{U}}^1_{\tau_i}$ have orthonormal columns for $i = 1, \dots, m$. We recall that $\widehat{C}^1_\tau$ is the solution at t_1 of the differential equation

$$\dot{\widehat{C}}_\tau(t) = F_\tau\left(t, \widehat{C}_\tau(t) \bigtimes_{i=1}^m \widehat{\mathbf{U}}^1_{\tau_i}\right) \bigtimes_{i=1}^m \widehat{\mathbf{U}}^{1,*}_{\tau_i}$$

with initial value $\widehat{C}_\tau(t_0) = \widehat{C}^0_\tau$. By a standard argument and using (4.15) we get

$$\begin{aligned}\frac{1}{2}\frac{d}{dt}\|\widehat{C}_\tau(t)\|^2 &= \operatorname{Re}\langle \widehat{C}_\tau(t), \dot{\widehat{C}}_\tau(t)\rangle = \operatorname{Re}\langle \widehat{C}_\tau(t), F_\tau(t, \widehat{C}_\tau(t) \bigtimes_{i=1}^m \widehat{\mathbf{U}}^1_{\tau_i}) \bigtimes_{i=1}^m \widehat{\mathbf{U}}^{1,*}_{\tau_i}\rangle \\ &= \operatorname{Re}\langle \widehat{C}_\tau(t) \bigtimes_{i=1}^m \widehat{\mathbf{U}}^1_{\tau_i}, F_\tau(t, \widehat{C}_\tau(t) \bigtimes_{i=1}^m \widehat{\mathbf{U}}^1_{\tau_i})\rangle = 0.\end{aligned}$$

This gives us

$$\|\widehat{Y}^1_\tau\| = \|\widehat{C}^1_\tau\| = \|\widehat{C}_\tau(t_1)\| = \|\widehat{C}_\tau(t_0)\| = \|\widehat{C}^0_\tau\| = \|\widehat{Y}^0_\tau\| = \|Y^0_\tau\|,$$

where the last equality comes from lemma 4.6. □

4.7.4 Energy preservation

We will now proof that the rank-adaptive BUG integrator also preserves the energy for Hamiltonian systems. For this consider the tensor Schrödinger equation

$$\mathrm{i}\dot{A}(t) = H[A(t)], \tag{4.16}$$

with a Hamiltonian $H : \mathbb{C}^{n_1 \times \cdots \times n_d} \to \mathbb{C}^{n_1 \times \cdots \times n_d}$ that is linear and self-adjoint, i.e., $\langle H[Y], Z\rangle = \langle Y, H[Z]\rangle$ for all $Y, Z \in \mathbb{C}^{n_1 \times \cdots \times n_d}$. The energy, defined by

$$\mathrm{E}(Y) = \langle Y, H[Y]\rangle,$$

is then preserved along solutions of (4.16), i.e. $\mathrm{E}(A(t)) = \mathrm{E}(A(t_0))$ for all t. The proof of the following theorem was originally presented in [CLS23, Theorem 6.3].

Theorem 4.5: Energy preservation

The rank-augmented TTN integrator preserves the energy: for every stepsize h,

$$\mathrm{E}(\widehat{Y}^1_{\bar\tau}) = \mathrm{E}(Y^0_{\bar\tau}).$$

Using the Cauchy–Schwarz inequality and theorem 2.4, this implies for the rank-adaptive TTN integrator

$$\begin{aligned}|\mathrm{E}(Y^1_{\bar\tau}) - \mathrm{E}(Y^0_{\bar\tau})| &= |\mathrm{E}(Y^1_{\bar\tau}) - \mathrm{E}(\widehat{Y}^1_{\bar\tau})| = |\mathrm{Re}\langle Y^1_{\bar\tau} - \widehat{Y}^1_{\bar\tau}, H[Y^1_{\bar\tau} + \widehat{Y}^1_{\bar\tau}]\rangle| \\ &\le c_{\bar\tau}\vartheta \, \|H[Y^1_{\bar\tau} + \widehat{Y}^1_{\bar\tau}]\|,\end{aligned}$$

which implies a energy preservation up to truncation tolerance.

Proof. Inserting the differential equation from algorithm 7 for $\bar\tau = (\tau_1, \ldots, \tau_m)$ and using $\widehat{Y}_{\bar\tau}(t) = \widehat{C}_{\bar\tau}(t) \bigtimes_{j=1}^m \widehat{\mathbf{U}}_{\tau_j}$, we obtain by deriving the energy of $\widehat{Y}_{\bar\tau}(t)$ in time

$$\begin{aligned}\frac{d}{dt}\mathrm{E}(\widehat{Y}_{\bar\tau}(t)) &= 2\,\mathrm{Re}\langle H[\widehat{Y}_{\bar\tau}(t)], \dot{\widehat{C}}_{\bar\tau}(t) \bigtimes_{j=1}^m \widehat{\mathbf{U}}^1_{\tau_j}\rangle \\ &= 2\,\mathrm{Re}\langle H[\widehat{Y}_\tau(t)] \bigtimes_{j=1}^m \widehat{\mathbf{U}}^{1,*}_{\tau_j}, \dot{\widehat{C}}_{\bar\tau}(t)\rangle \\ &= 2\,\mathrm{Re}\langle H[\widehat{Y}_\tau(t)] \bigtimes_{j=1}^m \widehat{\mathbf{U}}^{1,*}_{\tau_j}, -\mathrm{i}H[\widehat{Y}_{\bar\tau}(t)] \bigtimes_{j=1}^m \widehat{\mathbf{U}}^{1,*}_{\tau_j}\rangle \\ &= -2\,\mathrm{Re}\ \mathrm{i}\, \|H[\widehat{Y}_\tau(t)] \bigtimes_{i=j}^m \widehat{\mathbf{U}}^{1,*}_{\tau_j}\|^2 = 0.\end{aligned}$$

Using $\widehat{Y}^1_\tau = \widehat{Y}_\tau(t_1)$ and lemma 4.6, we obtain

$$\mathrm{E}(\widehat{Y}^1_\tau) = \mathrm{E}(\widehat{Y}_\tau(t_1)) = \mathrm{E}(\widehat{Y}_\tau(t_0)) = \mathrm{E}(\widehat{Y}^0_\tau) = \mathrm{E}(Y^0_\tau),$$

which proves result. □

For every subtree $\tau \le \bar\tau$, the reduced operator F_τ that corresponds to the full $F_{\bar\tau}(Y) = -\mathrm{i}H[Y]$ is again a Schrödinger operator, i.e. $F_\tau(Y_\tau) = -\mathrm{i}H_\tau[Y_\tau]$. Note that $H_\tau : \mathcal{V}_\tau \to \mathcal{V}_\tau$

is again a linear and self-adjoint operator since $H_{\tau_i} = \pi_{\tau,i}^{\dagger} H_\tau \pi_{\tau,i}$. Hence, the rank-adaptive BUG integrator for TTNs preserves the energy on each subtree.

4.7.5 Energy diminishing for gradient systems

Consider the case of a tensor differential equation, where the right-hand side is a gradient system, i.e.

$$\dot{A}(t) = -\nabla \mathrm{E}(A(t)) \quad \text{with a given function } \mathrm{E} : \mathbb{R}^{n_1 \times \cdots \times n_d} \to \mathbb{R}.$$

In this situation, along every solution we then have the energy decay

$$\frac{d}{dt}\mathrm{E}(A(t)) = \langle \nabla \mathrm{E}(A(t)), \dot{A}(t) \rangle = -\|\nabla \mathrm{E}(A(t))\|^2.$$

We will prove that the rank-adaptive BUG integrator also diminishes the energy for gradient systems. The following theorem and its proof can be found [CLS23, Theorem 6.4].

Theorem 4.6: Energy diminishing for gradient systems

The rank-augmented TTN integrator diminishes the energy: for every stepsize $h > 0$,

$$\mathrm{E}(\widehat{Y}_{\bar{\tau}}^1) \le \mathrm{E}(Y_{\bar{\tau}}^0) - \alpha^2 h,$$

where $\alpha = \min_{0 \le \mu \le 1} \|\nabla \mathrm{E}(\widehat{Y}_{\bar{\tau}}(t_0 + \mu h)) \times_{i=1}^m \widehat{\mathbf{U}}_{\tau_i}^{1,*}\| = \|\nabla \mathrm{E}(Y_{\bar{\tau}}^0) \times_{i=1}^m \widehat{\mathbf{U}}_{\tau_i}^{1,*}\| + O(h)$.

By the mean value theorem and theorem 2.4, this implies for the rank-adaptive BUG integrator for TTNs that

$$\mathrm{E}(Y_{\bar{\tau}}^1) \le \mathrm{E}(\widehat{Y}_{\bar{\tau}}^1) + \beta c_{\bar{\tau}} \vartheta \le \mathrm{E}(Y_{\bar{\tau}}^0) - \alpha^2 h + \beta c_{\bar{\tau}} \vartheta$$

with $\beta = \max_{0 \le \mu \le 1} \|\nabla \mathrm{E}(\mu Y_{\bar{\tau}}^1 + (1-\mu)\widehat{Y}_{\bar{\tau}}^1)\| = \|\nabla \mathrm{E}(Y_{\bar{\tau}}^1)\| + O(\vartheta)$. Thus, the rank-adaptive BUG reduces the energy if the truncation tolerance ϑ is chosen small enough.

Proof. We prove the result by a similar strategy as in the previous proof, by deriving the energy $\mathrm{E}(\widehat{Y}_{\bar{\tau}}(t))$ with respect to time. In algorithm 7 we have for $\bar{\tau} = (\tau_1, \ldots, \tau_m)$

and $\widehat{Y}_{\bar{\tau}}(t) = \widehat{C}_{\bar{\tau}}(t) \times_{i=1}^{m} \widehat{\mathbf{U}}^1_{\tau_i}$

$$\begin{aligned}
\frac{d}{dt}\mathrm{E}(\widehat{Y}_{\bar{\tau}}(t)) &= \langle \nabla \mathrm{E}(\widehat{Y}_{\bar{\tau}}(t)), \dot{\widehat{C}}_{\bar{\tau}}(t) \bigtimes_{i=1}^{m} \widehat{\mathbf{U}}^1_{\tau_i} \rangle \\
&= \langle \nabla \mathrm{E}(\widehat{Y}_{\bar{\tau}}(t)) \bigtimes_{i=1}^{m} \widehat{\mathbf{U}}^{1,*}_{\tau_i}, \dot{\widehat{C}}_{\bar{\tau}}(t) \rangle \\
&= \langle \nabla \mathrm{E}(\widehat{Y}_{\bar{\tau}}(t)) \bigtimes_{i=1}^{m} \widehat{\mathbf{U}}^{1,*}_{\tau_i}, -\nabla \mathrm{E}(\widehat{Y}_{\bar{\tau}}(t)) \bigtimes_{i=1}^{m} \widehat{\mathbf{U}}^{1,*}_{\tau_i} \rangle \\
&= -\| \nabla \mathrm{E}(\widehat{Y}_{\bar{\tau}}(t)) \bigtimes_{i=1}^{m} \widehat{\mathbf{U}}^{1,*}_{\tau_i} \|^2 \leq -\alpha^2,
\end{aligned}$$

with $\alpha = \min_{0 \leq \mu \leq 1} \| \nabla \mathrm{E}(\widehat{Y}_{\bar{\tau}}(t_0 + \mu h)) \times_{i=1}^{m} \widehat{\mathbf{U}}^{1,*}_{\tau_i} \|$. By lemma 4.6 we have $\widehat{Y}^0_{\bar{\tau}} = Y^0_{\bar{\tau}}$ and with $\widehat{Y}^1_{\bar{\tau}} = \widehat{Y}_{\bar{\tau}}(t_1)$, we get

$$\mathrm{E}(\widehat{Y}^1_{\bar{\tau}}) = \mathrm{E}(\widehat{Y}_{\bar{\tau}}(t_1)) \leq \mathrm{E}(\widehat{Y}_{\bar{\tau}}(t_0)) - \alpha^2 h = \mathrm{E}(Y^0_{\bar{\tau}}) - \alpha^2 h,$$

which concludes the proof. □

As before, we note that for each subtree $\tau \leq \bar{\tau}$, the reduced operator F_τ that corresponds to the full operator $F_{\bar{\tau}}(Y) = -\nabla \mathrm{E}[Y]$, is again a gradient $F_\tau(Y_\tau) = -\nabla \mathrm{E}_\tau[Y_\tau]$ with an energy function $\mathrm{E}_\tau : \mathcal{V}_\tau \to \mathbb{R}$. As it is again defined recursively by $\mathrm{E}_{\tau_i} = \pi^{\dagger}_{\tau,i} \circ \mathrm{E}_\tau \circ \pi_{\tau,i}$ for the ith subtree τ_i of τ, the integrator dissipates the energy on the level of each subtree.

Chapter 5

Parallel BUG integrator for Tucker Tensor and Tree Tensor Networks

This chapter relies mainly on the work "A parallel basis update and Galerkin integrator for tree tensor networks" by Gianluca Ceruti, Jonas Kusch, Christian Lubich and the author [CKLS24]. It extends the parallel BUG integrator for matrices from [CKL24] to Tucker tensors and tree tensor networks.

We first recall that the rank-adaptive BUG integrator for tree tensor networks from chapter 4 is defined recursively. On each level of the tree, the corresponding nodes can be updated in parallel. Still, a fully parallel implementation can become complicated due to the recursive definition of the integrator itself. This motivates to derive an integrator where all nodes of the tree can be updated fully in parallel, i.e. all matrix and tensor differential equations are solved in parallel. For large trees, as is, for example, the case in many examples from quantum physics, this allows for much more efficient computations. The novel parallel integrator is furthermore demanded to pertain the desired rank-adaptivity property of the integrator of chapter 4. The remainder of this chapter derives such a method for Tucker tensors and tree tensor networks. For the Tucker tensor and the tree tensor network formats, we will further prove a robust error bound.

5.1 Recap: Parallel BUG integrator for matrices

In this section we introduce a variant of the parallel BUG integrator for matrices from [CKL24]. Consider a matrix-valued ordinary differential equation

$$\dot{\boldsymbol{A}}(t) = \boldsymbol{F}(t, \boldsymbol{A}(t)), \quad \boldsymbol{A}(t_0) = \boldsymbol{A_0},$$

with $\boldsymbol{A}(t) \in \mathbb{C}^{n\times m}$ for all t and $\boldsymbol{F} : \mathbb{C}^{n\times m} \to \mathbb{C}^{n\times m}$. Again we approximate the solution on the low-rank manifold $\mathcal{M}_r$, described in subsection 2.5.1, by projecting the right-hand side onto the tangent space of the current approximation at time t in $\mathcal{M}_r$.

One time step from t_0 to $t_1 = t_0 + h$ of the modified parallel BUG integrator for matrices reads now as follows. Suppose initial data $\boldsymbol{Y}^0 = \mathbf{U}^0\mathbf{S}^0\mathbf{V}^{0,*}$ of rank r, with $\mathbf{U}^0$ and $\mathbf{V}^0$ orthonormal matrices. We compute the approximation $\boldsymbol{Y}^1 = \mathbf{U}^1\mathbf{S}^1\mathbf{V}^{1,*}$ at time t_1 by the following three steps:

1. Construct the augmented basis matrices $\widehat{\mathbf{U}}^1 \in \mathbb{C}^{n\times 2r}$ and $\widehat{\mathbf{V}}^1 \in \mathbb{C}^{m\times 2r}$ as well as the coefficient matrix $\bar{\mathbf{S}}(t_1) \in \mathbb{C}^{r\times r}$ (in parallel):

   K-step: From $t = t_0$ to t_1 integrate the $n \times r$ matrix differential equation

   $$\dot{\mathbf{K}}(t) = \boldsymbol{F}(t, \boldsymbol{K}(t)\mathbf{V}^{0,*})\mathbf{V}^0, \qquad \boldsymbol{K}(t_0) = \mathbf{U}^0\mathbf{S}^0. \tag{5.1}$$

   Construct $\widehat{\mathbf{U}}^1 = (\mathbf{U}^0, \widetilde{\mathbf{U}}^1) \in \mathbb{C}^{n\times 2r}$ as an orthonormal basis of the range of the $n\times 2r$ matrix $(\mathbf{U}^0, \mathbf{K}(t_1))$ (e.g. by QR decomposition).
   Compute the matrix $\widetilde{\mathbf{S}}^1_K = h\widetilde{\mathbf{U}}^{1,*}\boldsymbol{F}(\boldsymbol{Y}^0)\mathbf{V}^0 \in \mathbb{C}^{r\times r}$.

   L-step: From $t = t_0$ to t_1 integrate the $m \times r$ matrix differential equation

   $$\dot{\mathbf{L}}(t) = \boldsymbol{F}(t, \mathbf{U}^0\mathbf{L}(t)^*)^*\mathbf{U}^0, \qquad \mathbf{L}(t_0) = \mathbf{V}^0\mathbf{S}^{0,*}. \tag{5.2}$$

   Construct $\widehat{\mathbf{V}}^1 = (\mathbf{V}^0, \widetilde{\mathbf{V}}^1) \in \mathbb{C}^{m\times 2r}$ as an orthonormal basis of the range of the $m \times 2r$ matrix $(\mathbf{V}^0, \mathbf{L}(t_1))$ (e.g. by QR decomposition)
   Compute the matrix $\widetilde{\mathbf{S}}^1_L = h\mathbf{U}^{0,*}\boldsymbol{F}(\boldsymbol{Y}^0)^*\widetilde{\mathbf{V}}^1 \in \mathbb{C}^{r\times r}$.

   S-step: From $t = t_0$ to t_1 integrate the $r \times r$ matrix differential equation

   $$\dot{\bar{\mathbf{S}}}(t) = \mathbf{U}^{0,*}\boldsymbol{F}(t, \mathbf{U}^0\bar{\mathbf{S}}(t)\mathbf{V}^{0,*})\mathbf{V}^0, \qquad \bar{\mathbf{S}}(t_0) = \mathbf{S}^0. \tag{5.3}$$

2. **Augment**: Construct the augmented coefficient matrix $\widehat{\mathbf{S}}^1 \in \mathbb{C}^{2r\times 2r}$ as

   $$\widehat{\mathbf{S}}^1 = \begin{pmatrix} \bar{\mathbf{S}}(t_1) & \widetilde{\mathbf{S}}^1_L \\ \widetilde{\mathbf{S}}^1_K & \mathbf{0} \end{pmatrix}. \tag{5.4}$$

3. **Truncate**: Compute the singular value decomposition $\widehat{\mathbf{S}}^1 = \widehat{\mathbf{P}}\widehat{\mathbf{\Sigma}}\widehat{\mathbf{Q}}^*$ where $\widehat{\mathbf{\Sigma}} = \operatorname{diag}(\sigma_j)$. Truncate to the tolerance ϑ by choosing the new rank $r_1 \leq 2r$ as the minimal number r_1 such that

$$\left(\sum_{j=r_1+1}^{2r} \sigma_j^2 \right)^{1/2} \leq \vartheta. \tag{5.5}$$

   Compute the new factors for the approximation of $\boldsymbol{A}(t_1)$ as follows: Let $\mathbf{S}^1$ be the $r_1 \times r_1$ diagonal matrix with the r_1 largest singular values and let $\mathbf{P}^1 \in \mathbb{C}^{2r \times r_1}$ and $\boldsymbol{Q}^1 \in \mathbb{C}^{2r \times r_1}$ contain the first r_1 columns of $\widehat{\mathbf{P}}$ and $\widehat{\mathbf{Q}}$, respectively. Finally, set $\mathbf{U}^1 = \widehat{\mathbf{U}}^1\mathbf{P}^1 \in \mathbb{C}^{n \times r_1}$ and $\mathbf{V}^1 = \widehat{\mathbf{V}}^1\boldsymbol{Q}^1 \in \mathbb{C}^{m \times r_1}$. Note that this truncation strategy equals the truncation algorithm for tree tensor networks applied to matrices, which was discussed in subsection 2.4.4.

All differential equations are solvable in parallel, while the augmentation and truncation step have to be performed sequentially. However, the main computational complexity lies in the solution of the arising differential equations. Further, in the **S**-step only a $r \times r$ matrix ODE has to be solved, while in the **S**-step for the rank-adaptive BUG a $2r \times 2r$ matrix ODE must be solved, cf. [CKL22].
By the augmentation and truncation step 2. and 3., the integrator is rank-adaptive. At each time step with initial rank r, the integrator chooses a new rank $r_1 \leq 2r$. If in practice a higher rank increase per time step is needed, a step rejection strategy can be added, which will be discussed later, see subsection 5.3.3.

In [CKL24] the authors proved a robust error bound for the parallel BUG integrator for matrices, which is also shared by the rank-adaptive BUG, fixed-rank BUG and projector-splitting integrator for matrices [CKL22, CL21, LO14]. Robust means that all appearing constants do not depend on singular values of the matrix **S**, cf. [CKL24, Theorem 4.1]. The error bound is proven by interpreting the parallel BUG integrator for matrices as a perturbed rank-adaptive BUG integrator for matrices. The proof of the robust error bound for the Tucker version of the parallel BUG integrator will follow the same idea.

Remark 5.1. It is important to note that the orthogonalization of the matrices $\widehat{\mathbf{U}}^1$ and $\widehat{\mathbf{V}}^1$ must be done carefully. The orthogonalization must ensure that the first r columns of $\widehat{\mathbf{U}}^1$ equal $\mathbf{U}^0$. Analogously the first r columns of $\widehat{\mathbf{V}}^1$ must equal $\mathbf{V}^0$. This ordering of the old and new basis is not needed in the rank-adaptive BUG integrator presented in chapter 4.

Remark 5.2. The proposed version of the parallel BUG integrator for matrices differs from the variant in [CKL24] only in the definition of the matrices $\widetilde{\mathbf{S}}_k^1$ and $\widetilde{\mathbf{S}}_L^1$. Note that in [CKL24] they were defined by $\widetilde{\mathbf{S}}_K^1 = \widetilde{\mathbf{U}}^{1,*}\mathbf{K}(t_1)$ and $\widetilde{\mathbf{S}}_L^1 = \mathbf{L}(t_1)^*\widetilde{\mathbf{V}}^1$. Hence, in the variant from above, the original $\widetilde{\mathbf{S}}_K^1$ and $\widetilde{\mathbf{S}}_L^1$ have been replaced by a first-order

approximation:

$$\mathbf{S}_K^1 = \widetilde{\mathbf{U}}^{1,*}\mathbf{K}(t_1) = \widetilde{\mathbf{U}}^{1,*}\left(\mathbf{K}^0 + h\boldsymbol{F}(\mathbf{Y}^0)\mathbf{V}^0 + O(h^2)\right) = \widetilde{\mathbf{S}}_K^1 + O(h^2),$$

and analogously for $\widetilde{\mathbf{S}}_L^1$.

This minor modification enabled us to extend the integrator and its robust error bound in a natural way to first Tucker tensors and later tree tensor networks.

5.2 Parallel BUG integrator for Tucker Tensors

In this section we present a parallel BUG integrator for Tucker tensors, which can be applied to problems of the form

$$\dot{A}(t) = F(t, A(t)), \quad A(t_0) = A^0,$$

where $A(t) \in \mathbb{C}^{n_1 \times \cdots \times n_d}$. For $r = (r_1, \ldots, r_d)$, we approximate the solution on the low-rank manifold of tensors of multilinear rank r denoted by $\mathcal{M}_r$, as discussed in subsection 2.5.2. At time t_0 we suppose the initial data Y^0 in a Tucker decomposition

$$Y^0 = C^0 \bigtimes_{i=1}^{d} \mathbf{U}_i^0,$$

where $C^0 \in \mathbb{C}^{r_1 \times \cdots \times r_d}$ and $\mathbf{U}_i^0 \in \mathbb{C}^{n_i \times r_i}$ orthonormal for $i = 1, \ldots, d$. As in section 4.1 we define the matricization of $F(t, \cdot)$ by $\boldsymbol{F}_i(t, Y) := \mathbf{Mat}_i(F(t, Ten_i(Y))$ and $n_{\neg i} := \prod_{j \neq i} n_j$. In the following, we propose a parallel integrator for Tucker tensors, where the differential equations for the core tensor C^0 and all basis matrices $\mathbf{U}_i^0$ can be evolved fully in parallel.

5.2.1 Formulation of the algorithm

Given the initial data $Y^0 = C^0 \times_{i=1}^d \mathbf{U}_i^0$ of multilinear rank r at time t_0, we aim to find an approximation $Y^1 = C^1 \times_{i=1}^d \mathbf{U}_i^1$ at time t_1. Again, we make use of the subflow notation $\Phi^{(i)}$ and Ψ for the update of the ith basis matrix and core tensor, respectively. Note that we use the same subflow notation as in section 4.1, even if the subflows for the parallel BUG differ from the subflows of the rank-adaptive BUG. Additionally to the subflows, we apply an augmentation step $\mathcal{A}$ and a rank truncation Θ. The approximation Y^1 is then obtained by

$$Y^1 = \Theta \circ \mathcal{A} \circ (\Psi, \Phi^{(1)}, \ldots, \Phi^{(d)})(Y^0). \tag{5.6}$$

The full algorithm for the parallel BUG integrator for Tucker tensor then reads as

Algorithm 9: Parallel BUG integrator for Tucker tensors

Data: Tucker tensor $Y^0 = C^0 \times_{i=1}^d \mathbf{U}_i^0$ in factorized form of multilinear rank $(r_1^0, \dots, r_d^0)$, function $F(t, Y)$, t_0, t_1, tolerance parameter ϑ

Result: Tucker tensor $Y^1 = C^1 \times_{i=1}^d \mathbf{U}_i^1$ in factorized form of multilinear rank $(r_1^1, \dots, r_d^1)$, where $r_i^1 \leq 2r_i^0$

begin

  for $i = 1 : d$ in parallel **do**

    compute $[\widehat{\mathbf{U}}_i^1, \widetilde{C}_i^1] = \Phi^{(i)}\big(Y^0, F, t_0, t_1\big)$

    `% update and augment the` i`th basis matrix, see algorithm 10`

  end

  compute $\bar{C}^1 = \Psi\big(C^0, (\mathbf{U}_i^0)_{i=1}^d, F, t_0, t_1\big)$

  `% update the core tensor, see algorithm 11`

  construct $\widehat{C}^1 = \mathcal{A}\left(\bar{C}^1, (\widetilde{C}_i^1)_{i=1}^d\right)$

  `% augment the core tensor, see algorithm 12`

  set $\widehat{Y}^1 = \widehat{C}^1 \times_{i=1}^d \widehat{\mathbf{U}}_i^1$

  compute $Y^1 = \Theta\big(\widehat{Y}^1, \vartheta\big)$ `% rank truncation, see algorithm 1`

end

Note that the subflows $\Phi^{(i)}, i = 1, \dots, d$ and Ψ can be done fully in parallel. After these are solved, the augmentation of the core tensor and finally a rank truncation is performed sequentially.

Algorithm 10: Subflow $\Phi^{(i)}$ (update and augment the ith basis matrix))

Data: Tucker tensor $Y^0 = C^0 \times_{j=1}^{d} \mathbf{U}_j^0$ of multilinear rank $(r_1^0, \ldots, r_d^0)$ in factorized form, function $F(t, Y)$, t_0, t_1

Result: Updated and augmented basis matrix $\widehat{\mathbf{U}}_i^1 \in \mathbb{C}^{n_i \times \widehat{r}_i}$ (typically $\widehat{r}_i = 2r_i^0$) with orthonormal columns, tensor $\widetilde{C}_i^1 \in \mathbb{C}^{r_1^0 \times \cdots \times r_{i-1}^0 \times (\widehat{r}_i - r_i^0) \times r_{i+1}^0 \times \cdots \times r_d^0}$

begin

compute the QR-decomposition $\mathbf{Mat}_i(C^0)^\top = \mathbf{Q}_i^0 \mathbf{S}_i^{0,\top}$;

solve the $n_i \times r_i^0$ matrix differential equation from t_0 to t_1

$$\dot{\mathbf{K}}_i(t) = \mathbf{F}_i\big(t, \mathbf{K}_i(t)\mathbf{V}_i^{0,*}\big)\mathbf{V}_i^0, \qquad \mathbf{K}_i(t_0) = \mathbf{U}_i^0 \mathbf{S}_i^0,$$

with $\mathbf{F}_i(t, \cdot) := \mathbf{Mat}_i \circ F(t, \cdot) \circ Ten_i$ and $\mathbf{V}_i^{0,*} := \mathbf{Q}_i^\top \otimes_{j \neq i}^{d} \mathbf{U}_j^{0,\top}$;

construct $\widehat{\mathbf{U}}_i^1 = (\mathbf{U}_i^0, \widetilde{\mathbf{U}}_i^1) \in \mathbb{C}^{n_i \times \widehat{r}_i}$ as an orthonormal basis of the range of the $n_i \times 2r_i^0$ matrix $(\mathbf{U}_i^0, \mathbf{K}_i(t_1))$ (e.g. by QR decomposition) such that the first r_i^0 columns equal $\mathbf{U}_i^0$;

set $\widetilde{C}_i^1 = hF(Y_0) \times_{j \neq i} \mathbf{U}_i^{0,*} \times_i \widetilde{\mathbf{U}}_i^{1,*}$;

end

Algorithm 11: Subflow Ψ (update the core tensor)

Data: core tensor $C^0 \in \mathbb{C}^{r_1^0 \times \cdots \times r_d^0}$, basis matrices $\mathbf{U}_i^0 \in \mathbb{C}^{n_i \times r_i^0}$ with orthonormal columns, function $F(t, Y)$, t_0, t_1

Result: core tensor $\bar{C}^1 \in \mathbb{C}^{r_1^0 \times \cdots \times r_d^0}$

begin

solve the $r_1^0 \times \cdots \times r_d^0$ tensor differential equation from t_0 to t_1,

$$\dot{\bar{C}}(t) = F\Big(t, \bar{C}(t) \bigtimes_{\ell=1}^{d} \mathbf{U}_\ell^0\Big) \bigtimes_{j=1}^{d} \mathbf{U}_j^{0,*}, \qquad \bar{C}(t_0) = C^0.$$

set $\bar{C}^1 = \widehat{C}(t_1)$

end

Algorithm 12: Augmentation $\mathcal{A}$

Data: core tensor $\bar{C}^1 \in \mathbb{C}^{r_1^0 \times \cdots \times r_d^0}$, tensors $(\widetilde{C}_i^1)_{i=1}^d$

Result: core tensor $\widehat{C}^1 \in \mathbb{C}^{\widehat{r}_1 \times \cdots \times \widehat{r}_d}$

begin

set $\widehat{C}^1 = \bar{C}^1$

for $i = 1 : d$ **do**

set $\widehat{C}^1 = Ten_i \begin{pmatrix} \mathbf{Mat}_i(\widehat{C}^1) \\ \mathbf{Mat}_i(\widetilde{C}_i^1) \end{pmatrix}$;

end

end

Remark 5.3. The augmentation of the core tensor can be interpreted also in a different way. We construct the augmented core tensor $\widehat{C}^1 \in \mathbb{C}^{\widehat{r}_1 \times \cdots \times \widehat{r}_d}$ such that

$$\widehat{C}^1 \bigtimes_{j=1}^{d} \mathbf{U}_j^{0,*} \widehat{\mathbf{U}}_j = \bar{C}(t_1)\,, \tag{5.7}$$

$$\widehat{C}^1 \bigtimes_{j \neq i} \mathbf{U}_j^{0,*} \widehat{\mathbf{U}}_j^1 \times_i \widetilde{\mathbf{U}}_i^{1,*} \widehat{\mathbf{U}}_i^1 = \widetilde{C}_i^1 \quad \text{for } i = 1, \cdots, d\,. \tag{5.8}$$

All remaining blocks are set to zero. This is a mathematically equivalent formulation of how the augmented core tensor is constructed in algorithm 12.

Comparison with rank-adaptive BUG

Since the presented parallel BUG and the rank-adaptive BUG from chapter 4 seem to be very similar, we want to go into the subtle differences. The differential equation for the $\mathbf{K}_i$-step (subflow $\Phi^{(i)}$) is the same for both integrators. However, in the orthogonalization of the matrix $(\mathbf{U}_i^0, \mathbf{K}_i(t_1))$ there is a minor but important difference. For the rank-adaptive BUG, any orthogonal basis of the range of this matrix is a possible choice, while for the parallel BUG only the orthogonal bases, where the first r_i^0 columns equal $\mathbf{U}_i^0$, are admissible. For this recall remark 5.1 in the matrix case.

Further, the Ψ subflow, which is a Galerkin step, is performed in different bases. The parallel BUG performs the Galerkin step in the old basis $\mathbf{U}_i^0$, while the rank-adaptive BUG performs it in the old and new basis $\widehat{\mathbf{U}}_i^1$. Hence, the tensor ODE is larger for the rank-adaptive BUG, since the basis has a larger (typically doubled) rank. This makes it computationally more costly. However, by this, all basis matrices and core tensors are augmented on the fly in the rank-adaptive BUG. In contrast, the parallel BUG requires an additional augmentation step. During this augmentation step, some blocks in the augmented core tensor are set to zero. In the augmented core tensor of the rank-adaptive BUG, all entries are typically non-zero. In total, this makes the parallel BUG integrator a faster but rougher integration scheme.

5.2.2 Robust error bound for Tucker tensors

The presented parallel Tucker integrator from algorithm 9 shares the robust error bound of the rank-adaptive BUG integrator [CKL22] for Tucker tensors. Let $\mathcal{V} := \mathbb{C}^{n_1 \times \cdots \times n_d}$. To prove the robust error bound we make the following three assumptions:

1. $F : [0, t^*] \times \mathcal{V} \to \mathcal{V}$ is Lipschitz continuous and bounded, i.e.

$$\begin{aligned} \left\| F(t,Y) - F(t,\widetilde{Y}) \right\| &\leq L \left\| Y - \widetilde{Y} \right\| && \forall Y, \widetilde{Y} \in \mathcal{V} \\ \left\| F(t,Y) \right\| &\leq B && \forall Y \in \mathcal{V}. \end{aligned}$$

2. Given the projector onto the tangent space at $Y = C \bigtimes_{j=1}^{d} \mathbf{U}_j$ as

$$P(Y)Z := Z \bigtimes_{j=1}^{d} \mathbf{U}_j \mathbf{U}_j^* + \sum_{k=1}^{d} Z \bigtimes_{j \neq k} \mathbf{U}_j \mathbf{U}_j^* \times_k \left(\mathbf{I} - \mathbf{U}_k \mathbf{U}_k^* \right), \tag{5.9}$$

   cf. lemma 2.3, the normal component of $F(t, Y_k)$ fulfills $\| P(Y_k) F(t, Y_k) \| \leq \varepsilon$ for $0 \leq kh \leq T$.

3. The error at the initial condition is bounded by $\| Y_0 - A(0) \| \leq \delta$.

Then, the following error bound holds. The result can be found in [CKLS24, Section 3.2].

Theorem 5.1: Robust error bound for parallel BUG for Tucker tensors

Under assumptions 1. to 3., the error of the parallel BUG integrator for Tucker tensors is bounded by

$$\| Y_k - A(t_k) \| \leq c_0 \delta + c_1 h + c_2 \varepsilon + c_3 k \vartheta,$$

where all arising constants only depend on the bound and Lipschitz constant of F and on t_k.

Proof.
With $\widehat{Y}_{\mathrm{BUG}}^1$ we denote the approximation at time t_1 obtained from the rank-adaptive BUG integrator for Tucker tenors. By the triangle inequality, the local error of the parallel BUG approximation can be bounded by

$$\| \widehat{Y}_1 - A(t_1) \| \leq \| \widehat{Y}_1 - \widehat{Y}_{\mathrm{BUG}}^1 \| + \| \widehat{Y}_{\mathrm{BUG}}^1 - A(t_1) \|.$$

By [CKL22], we know that the rank-adaptive BUG integrator for Tucker tensors fulfils $\|\widehat{Y}^1_{\mathrm{BUG}} - A(t_1)\| \leq c_1 h^2 + c_2 h\varepsilon$. Thus, we only need to bound

$$\|\widehat{Y}_1 - \widehat{Y}^1_{\mathrm{BUG}}\| = \|\widehat{C}^1 - \widehat{C}^1_{\mathrm{BUG}}\|\,.$$

We now investigate the norms of individual blocks in $\widehat{C}^1 - \widehat{C}^1_{\mathrm{BUG}}$. To investigate the error between the core tensors $\widehat{C}^1$ and $\widehat{C}_{\mathrm{BUG}}$, we investigate three parts: The block that belongs to (5.7), blocks belonging to (5.8), and blocks that are set to zero.

1. First, we investigate the local error of the parallel integrator in the block that belongs to (5.7). Therefore, we define $\bar{Y}(t) \coloneqq \bar{C}(t)\times_{i=1}^d \mathbf{U}_i^0$, $\widehat{Y}_{\mathrm{BUG}}(t) \coloneqq \widehat{C}_{\mathrm{BUG}}(t)\times_{i=1}^d \widehat{\mathbf{U}}_i$, and $\widehat{F}(t) \coloneqq F(t, \widehat{Y}_{\mathrm{BUG}}(t))$. We note that

$$\begin{aligned}\|\widehat{Y}_{\mathrm{BUG}}(t) - \bar{Y}(t)\| &= \|\int_{t_0}^t \widehat{F}(s) \bigtimes_{i=1}^d \widehat{\mathbf{U}}_i^1 \widehat{\mathbf{U}}_i^{1,*}\, ds - \int_{t_0}^t F(s, \bar{Y}(s)) \bigtimes_{i=1}^d \mathbf{U}_i^0 \mathbf{U}_i^{0,*}\, ds\| \\ &\leq \int_{t_0}^t \|\widehat{F}(s)\| + \|F(s, \bar{Y}(s))\|\, ds \\ &\leq 2B \int_{t_0}^t 1\, ds \leq 2Bh.\end{aligned}$$

   Using this we can bound the first block by

$$\begin{aligned}\|(\widehat{C}^1_{\mathrm{BUG}} - \widehat{C}^1) \bigtimes_{i=1}^d \mathbf{U}_i^{0,*} \widehat{\mathbf{U}}_i^1\| &\leq \int_{t_0}^{t_1} \|\widehat{F}(t) \bigtimes_{i=1}^d \mathbf{U}_i^{0,*} \widehat{\mathbf{U}}_i^1 \widehat{\mathbf{U}}_i^{1,*} - F(t, \bar{Y}(t)) \bigtimes_{i=1}^d \mathbf{U}_i^{0,*}\| \, dt \\ &= \int_{t_0}^{t_1} \|(\widehat{F}(t) - F(t, \bar{Y}(t))) \bigtimes_{i=1}^d \mathbf{U}_i^{0,*}\| \, dt \\ &\leq L \int_{t_0}^{t_1} \|\widehat{Y}_{\mathrm{BUG}}(t) - \bar{Y}(t)\| \, dt \leq cLBh^2\,.\end{aligned}$$

2. Second, we bound the local error of the integrator in the blocks that belong to (5.8). Here we first note that

$$\begin{aligned}\|\ \widehat{Y}_{\mathrm{BUG}}(t) - Y_0\| &= \|Y_0 + \int_{t_0}^t F(s, \widehat{Y}_{\mathrm{BUG}}(s)) \bigtimes_{i=1}^d \widehat{\mathbf{U}}_i^1 \widehat{\mathbf{U}}_i^{1,*} ds - Y_0\| \\ &\leq \int_{t_0}^t \|\ F(s, \widehat{Y}_{\mathrm{BUG}}(s)) \bigtimes_{i=1}^d \widehat{\mathbf{U}}_i^1 \widehat{\mathbf{U}}_i^{1,*}\| \, ds \leq Bh, \end{aligned} \tag{5.10}$$

   where the last bound follows by the assumptions. Using (5.10), we then obtain for the corresponding block:

$$\begin{aligned}\|(\widehat{C}^1_{\mathrm{BUG}} - \widehat{C}^1) \bigtimes_{j\neq i} \mathbf{U}_j^{0,*} \widehat{\mathbf{U}}_j^1 \times_i \widetilde{\mathbf{U}}_i^{1,*} \widehat{\mathbf{U}}_i^1\| &\leq \int_{t_0}^{t_1} \|(\widehat{F}(t) - F(Y_0)) \bigtimes_{j\neq i} \mathbf{U}_i^{0,*} \times_i \widetilde{\mathbf{U}}_i^{1,*}\| \, dt \\ &\leq L \int_{t_0}^{t_1} \|\widehat{Y}_{\mathrm{BUG}}(t) - Y_0\| \, dt \leq LBh^2\,.\end{aligned}$$

3. All remaining terms in $\widehat{C}^1$ are zero-valued. Hence, for any sets of indices $\mathcal{I} \subset \{1, \cdots, d\}$ with $|\mathcal{I}| \geq 2$ and $\mathcal{J} := \{1, \cdots, d\} \backslash \mathcal{I}$, we wish to bound the terms

$$\circledast := \|\widehat{C}^1_{\mathrm{BUG}} \bigtimes_{j \in \mathcal{J}} \mathbf{U}_j^{0,*} \widehat{\mathbf{U}}_j^1 \bigtimes_{i \in \mathcal{I}} \widetilde{\mathbf{U}}_i^{1,*} \widehat{\mathbf{U}}_i^1 \| \leq \int_{t_0}^{t_1} \|\widehat{F}(t) \bigtimes_{j \in \mathcal{J}} \mathbf{U}_j^{0,*} \bigtimes_{i \in \mathcal{I}} \widetilde{\mathbf{U}}_i^{1,*}\| \, dt \, .$$

From the definition of the projector onto the tangent space (5.9) we know that $P(Y_0)\widehat{F}(t_0) \times_{i \in \mathcal{I}} \widetilde{\mathbf{U}}_i^{1,*} = 0$ and therefore

$$\begin{aligned}
\circledast &\leq \int_{t_0}^{t_1} \|(\widehat{F}(t) - P(Y_0)\widehat{F}(t_0)) \bigtimes_{j \in \mathcal{J}} \mathbf{U}_j^{0,*} \bigtimes_{i \in \mathcal{I}} \widetilde{\mathbf{U}}_i^{1,*}\| \, dt \\
&\leq \int_{t_0}^{t_1} \|(\widehat{F}(t) - \widehat{F}(t_0)) \bigtimes_{j \in \mathcal{J}} \mathbf{U}_j^{0,*} \bigtimes_{i \in \mathcal{I}} \widetilde{\mathbf{U}}_i^{1,*}\| \, dt + h\varepsilon \\
&\leq \int_{t_0}^{t_1} \|\widehat{F}(t) - \widehat{F}(t_0)\| \, dt + h\varepsilon \leq L \int_{t_0}^{t_1} \|\widehat{Y}_{\mathrm{BUG}}(t) - Y_0\| \, dt + h\varepsilon \leq LBh^2 + h\varepsilon \, ,
\end{aligned}$$

where the last inequality is again due to (5.10).

Since $\|\widehat{Y}_1 - Y_1\| \leq \vartheta$, the local error becomes $\|Y_1 - A(t_1)\| \leq c_1 h^2 + c_2 h\varepsilon + c_3 \vartheta$. We then pass to the global error through Lady Windermere's fan [HNW93, Section I.7 and II.3], which concludes the proof. □

Remark 5.4. Note that many of the favourable properties of the rank-adaptive BUG integrator from chapter 4 are not valid anymore for the parallel BUG integrator. Namely, the exactness property, norm and energy preservation and the energy diminishing for gradient systems. We refer to chapter 6 for further numerical examples testing the integrators on these properties.

5.3 Parallel BUG integrator for Tree Tensor Networks

5.3.1 Formulation of the algorithm

Fix a tree $\bar{\tau} = (\bar{\tau}_1, \ldots, \bar{\tau}_m)$ and consider a corresponding tree tensor network

$$Y_{\bar{\tau}}^0 = C_{\bar{\tau}} \times_0 \mathbf{I} \bigtimes_{i=1}^{m} \mathbf{U}_{\bar{\tau}_i}^0 .$$

To evolve $Y_{\bar{\tau}}^0$ in time, on must update all basis matrices and all core tensors. The rank-adaptive BUG integrator for TTNs from chapter 4 allows the evolution of all basis matrices/core tensors on the same level in parallel. The parallel BUG integrator for TTNs (algorithm 13) allows the evolution of all basis matrices via the subflow Φ_l (algorithm 14) and all core tensors via the subflow Ψ_τ (algorithm 15) fully in parallel. Hence,

the computationally costly part of solving ordinary differential equations for each node can now be executed in parallel. To enable rank-adaptivity, a sequential and recursive augmentation step (algorithm 16), followed by a rank truncation (algorithm 1), is performed after solving all differential equations. Algorithm 16 is the natural extension of the augmentation step from algorithm 12 from Tucker tensors to tree tensor networks. When working with large tree tensor networks, the parallel BUG integrator enables high-performance computations through its fully parallel structure. For example, each differential equation in a time step of the algorithm can be solved on a different node on a cluster.

Algorithm 13: Parallel TTN BUG

Data: tree $\bar{\tau} = (\bar{\tau}_1, \dots, \bar{\tau}_m)$, TTN $Y^0_{\bar{\tau}} = C^0_{\bar{\tau}} \times_0 \mathbf{I} \times_{i=1}^m \mathbf{U}^0_{\bar{\tau}_i}$ in factorized form with tree ranks $(r^0_\tau)_{\tau \le \bar{\tau}}$, functions $(F_\tau(t,\cdot))_{\tau \le \bar{\tau}}$, t_0, t_1, truncation tolerance ϑ

Result: TTN $Y^1_{\bar{\tau}} = C^1_{\bar{\tau}} \times_0 \mathbf{I} \times_{i=1}^m \mathbf{U}^1_{\bar{\tau}_i}$ in factorized form with tree ranks $(r^1_\tau)_{\tau \le \bar{\tau}}$

begin

- **for** $\tau \le \bar{\tau}$ *in parallel* **do**
  - **if** $\tau = l$ *is a leaf* **then**
    - Set $\sigma = (\sigma_1, \dots, \sigma_m)$ such that $\sigma_i = l$ for an $i \in \{1, \dots, m\}$
    - `% Find the parent node in the tree`
    - Set $\widehat{\mathbf{U}}^1_\tau = \Phi_l(\sigma, l, C^0_\sigma, \mathbf{U}^0_l, F_l, t_0, t_1)$
    - `% Update the basis matrices, see algorithm 14`
  - **else**
    - Set $\bar{C}^1_\tau = \Psi_\tau(\tau, Y^0_\tau, F_\tau, t_0, t_1)$
    - `% Update the core tensors, see algorithm 15`
  - **end**
- **end**
- Set $\widehat{Y}^1_\tau = \mathcal{A}_{\bar{\tau}}(\bar{\tau}, (\bar{C}^1_\tau)_{\tau \le \bar{\tau}}, Y^0_\tau, (\widehat{\mathbf{U}}^1_l)_{l \in \mathcal{L}}, (F_\tau(t,\cdot))_{\tau \le \bar{\tau}}, h)$
- `% Augment the updated TTN, see algorithm 16`
- Set $Y^1_\tau = \Theta(\widehat{Y}^1_\tau, \vartheta)$
- `% Truncation with tolerance` ϑ`, see algorithm 1.`

end

5.3.1.1 Update of basis matrices and core tensors

Algorithm 14: Subflow Φ_l (update a basis matrix)

Data: tree $\tau = (\tau_1, \ldots, \tau_m)$ with $\tau_i = l$, C_τ^0 core tensor directly connected to $\mathbf{U}_l^0$, function $\boldsymbol{F}_l(t, \cdot)$, t_0, t_1

Result: $\widehat{\mathbf{U}}_l^1 = [\mathbf{U}_l^0, \widetilde{\mathbf{U}}_l^1]$

begin

compute a QR-decomposition $\mathbf{Mat}_i(C_\tau^0)^\top = \boldsymbol{Q}_l^0 \mathbf{S}_l^{0,\top}$;

set $\mathbf{Y}_l^0 = \mathbf{U}_l^{0,\top} \times_0 \mathbf{S}_l^{0,\top}$

solve the $n_l \times r_l^0$ matrix differential equation

$$\dot{\mathbf{Y}}_l(t) = \boldsymbol{F}_l(t, \mathbf{Y}_l(t)), \quad \mathbf{Y}_l(t_0) = \mathbf{Y}_l^0 \in \mathbb{C}^{r_l^0 \times n_l};$$

compute $\widehat{\mathbf{U}}_l^1 \in \mathbb{C}^{n_l \times \widehat{r}_l}$ with $\widehat{r}_l \leq 2r_l^0$ as an orthonormal basis of the range of the $n_l \times 2r_l^0$ matrix $\left(\mathbf{U}_l^0, \mathbf{Y}_l(t_1)^\top\right)$ such that the first r_l^0 columns of $\widehat{\mathbf{U}}_l^1$ equal $\mathbf{U}_l^0$;

end

Algorithm 15: Subflow Ψ_τ (update the core tensor)

Data: tree $\tau = (\tau_1, \ldots, \tau_m)$, core tensor $C_\tau^0 \in \mathbb{C}^{r_\tau^0 \times r_{\tau_1}^0 \times \cdots \times r_{\tau_m}^0}$, basis matrices $\mathbf{U}_{\tau_i}^0$ in factorized form such that $Y_\tau^0 = C_\tau^0 \times_0 \mathbf{I} \times_{i=1}^m \mathbf{U}_{\tau_i}^0$, function $F_\tau(t, \cdot)$, t_0, t_1

Result: core tensors $\bar{C}_\tau^1 \in \mathbb{C}^{r_\tau^0 \times r_{\tau_1}^0 \times \cdots \times r_{\tau_m}^0}$

begin

solve the $r_\tau^0 \times r_{\tau_1}^0 \times \cdots \times r_{\tau_m}^0$ tensor differential equation from t_0 to t_1

$$\dot{\bar{C}}_\tau(t) = F_\tau\Big(t, \bar{C}_\tau(t) \bigtimes_{i=1}^m \mathbf{U}_{\tau_i}^0\Big) \bigtimes_{i=1}^m \mathbf{U}_{\tau_i}^{0,*}, \quad \bar{C}_\tau(t_0) = C_\tau^0;$$

set $\bar{C}_\tau^1 = \bar{C}_\tau(t_1)$

end

5.3.1.2 Augmentation

Algorithm 16: Augmentation $\mathcal{A}_\tau$

Data: tree $\tau = (\tau_1, \dots, \tau_m)$, core tensor $\bar{C}^1_\tau \in \mathbb{C}^{r^0_\tau \times r^0_{\tau_1} \times \cdots \times r^0_{\tau_m}}$, core tensors $(\bar{C}^1_\varsigma)_{\varsigma<\tau}$, tree tensor network Y^0_τ, basis matrices $\widehat{\mathbf{U}}^1_l$ for $l \in \mathcal{L}$, functions $(F_\varsigma(t,\cdot))_{\varsigma\leq\tau}$, time step size h

Result: augmented TTN $\widehat{Y}^1_\tau = \widehat{C}^1_\tau \times_0 \mathbf{I} \bigtimes_{i=1}^m \widehat{\mathbf{U}}^1_{\tau_i}$ in factorized form of tree rank $(\widehat{r}^1_\varsigma)_{\varsigma\leq\tau}$ with $\widehat{r}^1_\varsigma \leq 2r^0_\varsigma$

begin

% Augmentation of subtrees

for $i = 1 : m$ **do**

if $\tau_i \notin \mathcal{L}$, *i.e.* $\tau_i = (\sigma_1, \dots, \sigma_k)$ **then**

$\widehat{Y}^1_{\tau_i} = \mathit{Augmentation}(\tau_i, (\bar{C}^1_\varsigma)_{\varsigma\leq\tau_i}, Y^0_{\tau_i}, (\widehat{\mathbf{U}}^1_l)_{l\in\mathcal{L}}, (F_\varsigma(t,\cdot))_{\varsigma\leq\tau_i}, h)$

set $\widehat{C}^0_{\tau_i} = C^0_{\tau_i} \bigtimes_{j=1}^k \widehat{\mathbf{U}}^{1,*}_{\sigma_j} \mathbf{U}^0_{\sigma_j}$

compute an orthonormal basis $\widehat{\boldsymbol{Q}}_{\tau_i}$ of the range of $(\mathbf{Mat}_0(\widehat{C}^0_{\tau_i})^\top, \mathbf{Mat}_0(\widehat{C}^1_{\tau_i})^\top)$ such that the first $r^0_{\tau_i}$ columns of $\widehat{\boldsymbol{Q}}_{\tau_i}$ equal $\mathbf{Mat}_0(\widehat{C}^0_{\tau_i})^\top$;

set $\widehat{\mathbf{U}}^1_{\tau_i} = \mathbf{Mat}_0(\widehat{X}^1_{\tau_i})^\top$, where the orthonormal TTN $\widehat{X}^1_{\tau_i}$ is obtained from $\widehat{Y}^1_{\tau_i}$ by replacing the core tensor with $\widehat{C}^1_{\tau_i} = \mathit{Ten}_0(\widehat{\boldsymbol{Q}}^\top_{\tau_i})$;

% Construction of $\widetilde{\mathbf{U}}^1_{\tau_i}$

set $\widehat{r}^1_{\tau_i}$ as the number of columns of $\widehat{\boldsymbol{Q}}_{\tau_i}$

set $\boldsymbol{M}_{\tau_i}$ as the last $\widehat{r}^1_{\tau_i} - r^0_{\tau_i}$ columns of $\widehat{\boldsymbol{Q}}_{\tau_i}$

set $\widetilde{\mathbf{U}}^1_{\tau_i} = \mathbf{Mat}_0(\widetilde{X}^1_{\tau_i})^\top$, where the TTN $\widetilde{X}^1_{\tau_i}$ consists of the core tensor $\mathit{Ten}_0(\boldsymbol{M}^\top_{\tau_i})$ and $\widetilde{\mathbf{U}}^1_{\sigma_j} = \widehat{\mathbf{U}}^1_{\sigma_j}$, for $j = 1, \dots, k$;

end

% Augmentation of core tensor

Set $\widehat{C}^1_\tau = \bar{C}^1_\tau$

for $i = 1 : m$ **do**

if $\widetilde{\mathbf{U}}^1_{\tau_i}$ *is non-empty* **then**

compute the tensor $\widetilde{C}^1_i = hF(Y^0_\tau) \bigtimes_{j\neq i} \mathbf{U}^{0,*}_{\tau_j} \times_i \widetilde{\mathbf{U}}^{1,*}_{\tau_i}$

set $\widehat{C}^1_\tau = \mathit{Ten}_i \begin{pmatrix} \mathbf{Mat}_i(\widehat{C}^1_\tau) \\ \mathbf{Mat}_i(\widetilde{C}^1_i) \end{pmatrix}$

end

end

To give more intuition about the augmentation step, we provide a graphical illustration. In the subflow $\mathcal{A}_\tau$, each core tensor $\bar{C}^1_\tau$ is sequentially augmented in the ith mode by a block $\widetilde{C}^1_i$. This procedure is illustrated for a core tensor of a binary tree, i.e. an order three tensor, in the left part of figure 5.1. The augmentation in the 0-dimension (except for the root tensor) is performed after all subtrees have been augmented, due to

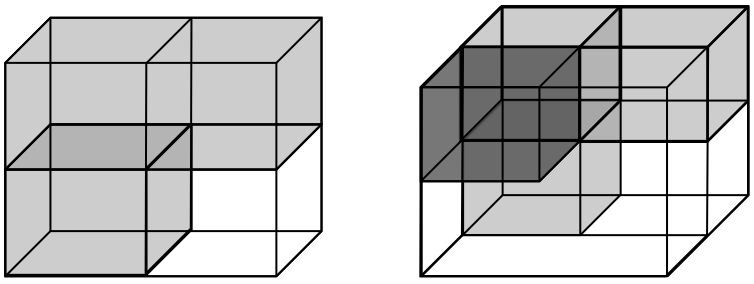

FIGURE 5.1: Augmentation of an order three tensor. Left: Illustration of the augmentation of a core tensor $\tilde{C}_\tau^1$ (light grey left top) with $\widetilde{C}_1$ in the first dimension (light grey left down) and with $\widetilde{C}_2$ in the second dimension (light grey right top). Right: Illustration of the augmentation of the core tensor from the left in the 0-dimension (dark grey block). All remaining blocks are set to zero.

the recursive structure. The 0-dimension augmentation is illustrated in the right part of figure 5.1. All the remaining blocks are set to zero, which is why the parallel BUG integrator gives a rougher approximation than the rank-adaptive BUG integrator, where all entries of the augmented core tensor are (usually) non-zero. We refer to the numerical example section for a more detailed comparison. Further, we want to emphasize that the augmentation step from algorithm 16 is a recursive process going from the bottom to the root of the tree.

5.3.2 Robust error bound for tree tensor networks

The robust error bound for the parallel BUG integrator for Tucker tensors from theorem 5.1 extends to the parallel BUG integrator for tree tensors networks. For a tree $\tau = (\tau_1, \dots, \tau_m)$, recall from subsection 4.7.2 the tensor space $\mathcal{V}_\tau = \mathbb{C}^{r_\tau \times n_{\tau_1} \times \cdots \times n_{\tau_m}}$ and the manifold of tree tensor networks at the kth time step $\mathcal{M}_\tau^k = \mathcal{M}((n_\tau)_{\tau \leq \bar{\tau}}, (r_\tau^k)_{\tau \leq \bar{\tau}})$. The manifold $\mathcal{M}_\tau^k$ has the additional index k due to the changing ranks in each time step. As before we set $\mathcal{M}^k = \mathcal{M}_{\bar{\tau}}^k$ and $\mathcal{V} = \mathcal{V}_{\bar{\tau}}$ for the full tree $\bar{\tau}$.
Following [CLS23, CLW21] and chapter 4, we make the following three assumptions

1. $F : [0, t^*] \times \mathcal{V}_{\bar{\tau}} \to \mathcal{V}_{\bar{\tau}}$ is Lipschitz continuous and bounded, i.e.

$$\begin{aligned} \left\| F(t, Y) - F(t, \widetilde{Y}) \right\| &\leq L \left\| Y - \widetilde{Y} \right\| && \forall\ Y, \widetilde{Y} \in \mathcal{V}_{\bar{\tau}} \\ \| F(t, Y) \| &\leq B && \forall\ Y \in \mathcal{V}_{\bar{\tau}}. \end{aligned}$$

2. For Y near the exact solution $A(t)$ and P_Y being the orthogonal projection onto the tangent space $\mathcal{T}_Y \mathcal{M}_{\bar{\tau}}^k$ we assume for all $t \in [t_k, t_{k+1}]$ the existence of a small $\epsilon > 0$ such that

$$\| F(t, Y) - P_Y F(t, Y) \| \leq \epsilon.$$

3. The error at the initial condition is bounded by $\|Y_0 - A(0)\| \leq \delta$.

This gives us the following error bound which can be found in [CKLS24, Section 4.7].

Theorem 5.2: Robust error bound for parallel BUG

Under the assumptions (1.-3.) from above, the error of the numerical solution after k time steps with the parallel BUG integrator for tree tensor networks can be bounded by

$$\left\|Y^k - A(t_k)\right\| \leq c_0\delta + c_1\epsilon + c_2 h + c_3 k\vartheta, \tag{5.11}$$

where all constants c_i are independent of the singular values of matricizations of core tensors.

We omit a detailed proof of theorem 5.2 as it goes exactly in the same way as the proof of theorem 4.2. By induction over the height of the tree, one uses the robust error bound for Tucker tensors from theorem 5.1 to prove the error bound for tree tensor networks.

5.3.3 A fully parallel step rejection strategy for binary trees

For some problems in applications, the solution to an ODE requires a sharp increase of the ranks at one time step [HS23]. In this situation, both the parallel BUG and the rank-adaptive BUG, fail to accurately capture the dynamic, as they only allow for a doubling of the tree ranks in each time step. To overcome this issue, the introduction of a step rejection strategy, which allows for an arbitrary increase of the ranks at each time step, is favourable. In this subsection we will extend the step-rejection strategy for matrices from [CKL24, Section 3.3] to binary tree tensor networks, see originally in [CKLS24, Section 4.6].
Suppose we have a TTN $Y^0_{\bar{\tau}}$ with ranks $(r^0_\tau)_{\tau\leq\bar{\tau}}$ at time t_0 and the TTN $\widehat{Y}^1_{\bar{\tau}}$ with ranks $(\widehat{r}^1_\tau)_{\tau\leq\bar{\tau}}$ at time t_1. For each subtree $\tau \leq \bar{\tau}$ check the following two conditions:

1) If $\widehat{r}^1_\tau = 2r^0_\tau$ for some $\tau < \bar{\tau}$, then the step is rejected.

2) If for some $\tau \leq \bar{\tau}$ the condition $h\eta_\tau > c\vartheta$ is satisfied (e.g. $c = 10$), where $\eta_\tau = \|F_\tau(Y^0_\tau) \times_1 \widetilde{\mathbf{U}}^{1,*}_{\tau_1} \times_2 \widetilde{\mathbf{U}}^{1,*}_{\tau_2}\|$ and $\widetilde{\mathbf{U}}^1_{\tau_i}$ for $i = 1, 2$ is defined as in algorithm 16, then the step is rejected. Note that all η_τ can be computed fully in parallel.

If a step is rejected, repeat the step with the augmented basis $\widehat{\mathbf{U}}^1_i$ for $i = 1, \ldots, d$ and augmented core tensors C^{aug}_τ, for $\tau \leq \bar{\tau}$, where C^{aug}_τ equals C^0_τ augmented with zeros such that its dimensions equal $(\widehat{r}_\tau, \widehat{r}_{\tau_1}, \ldots, \widehat{r}_{\tau_m})$.

The same strategy can be extended to non-binary trees. However, the number of step rejection constraints per node, i.e., the number of possible combinations of products with two or more $\widetilde{\mathbf{U}}^1_{\tau_i}$ factors and $\mathbf{U}^0_{\tau_i}$ as the remaining factors, scales geometrically as 2^{m-1}. Thus, if the order of the core tensors becomes larger, the step rejection strategy can become computationally expensive.

Remark 5.5. The step rejection strategy can be equally applied to the rank-adaptive BUG integrator for tree tensor networks from chapter 4.

Chapter 6

Numerical experiments

In this chapter we will apply the derived integrators for tree tensor networks from chapter 4 and 5 to various problems. Most of the examples are related to quantum physics. The first example is a synthetic one to verify the exactness property of the rank-adaptive and the fixed-rank BUG integrator. The remaining examples consider unitary and open quantum spin systems and the Schrödinger equation with a Henon-Heiles potential, a challenging problem from quantum molecular dynamics.
All experiments where done in Matlab using the Tensor Toolbox [BK+15], Tensorlab 3.0 [VDS+16] and the hm-toolbox [MRK20]. All simulations used a balanced binary tree unless explicitly stated otherwise. All code is provided online on GitHub [Sul24].

6.1 Exactness property

In this section we numerically verify the exactness property from theorem 4.1 for the fixed-rank BUG integrator and the rank-adaptive BUG integrator for tree tensor networks. Further, we will see that the parallel integrator for tree tensor networks from chapter 5 does not share the exactness property. The following example was taken from [CLW21], where the exactness property was verified for the projector-splitting integrator for tree tensor networks.

We fix a tree $\bar{\tau}$ with six leaves and the tree structure as in figure 6.1. For $\bar{\tau}$ consider a tree tensor network $X_{\bar{\tau}}^0 \in \mathcal{M}_{\bar{\tau}}$, with $n_l = 1000$ for all $l \in L(\bar{\tau})$, $r_\tau = 10$ for all $\tau \leq \bar{\tau}$ and randomly generated complex basis matrices and core tensors. Further, consider the time-dependent tree tensor network $X(t) = X_{\bar{\tau}}(t) \in \mathcal{M}_{\bar{\tau}}$ which is defined as follows. Let $\boldsymbol{W}_\tau \in \mathbb{R}^{r_\tau \times r_\tau}$ be a skew-symmetric matrix with $\|\boldsymbol{W}_\tau\|_F = 1$, for all $\tau \leq \bar{\tau}$. Then $X(t)$ is

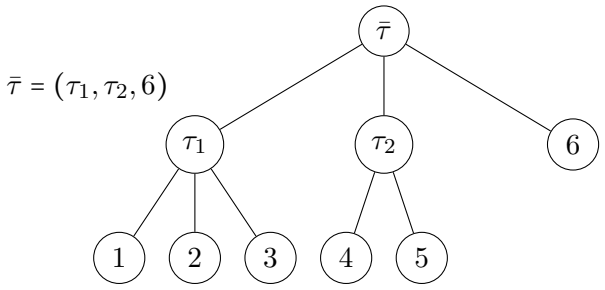

FIGURE 6.1: Tree structure for verifying the exactness property.

defined by

$$\begin{aligned} \mathbf{U}_l(t) &:= \mathbf{U}_l^0 e^{t\boldsymbol{W}_l}, && \text{for all } l \in L(\bar{\tau}), \\ C_\tau(t) &:= C_\tau^0 \times_0 e^{t\boldsymbol{W}_\tau}, && \text{for all } \tau \leq \bar{\tau} \text{ with } \tau \notin L(\bar{\tau}), \end{aligned}$$

where $\mathbf{U}_l^0$ with $l \in L(\bar{\tau})$ and C_τ^0 with $\tau \neq l$ are the leaves and core tensors of $X_{\bar{\tau}}^0$, respectively. Hence, $X(t)$ is of rank $r_\tau = 10$ for all $\tau \leq \bar{\tau}$ and all times $t \in [0,1]$, i.e. $X(t) \in \mathcal{M}_{\bar{\tau}}$. From this follows that the rank-adaptive BUG integrator and the fixed-rank BUG integrator for tree tensor networks can reproduce $X(t)$ exactly up to round-off errors.

We formulate the problem as an ordinary differential equation. The continuous problem reads

$$\dot{A}(t) = \dot{X}(t), \quad \text{with } A(0) = X(0).$$

We can reformulate this problem in an other way. Suppose we already have the approximation Y_n at time t_n to $X(t_n)$. Set $\Delta X_n = X(t_{n+1}) - X(t_n)$. Then we can compute the approximation X_{n+1} at time $t_{n+1} = t_n + h$ by performing one time step with an integrator with step size $\widetilde{h} = 1$ of the ordinary differential equation

$$\dot{X}(t) = \Delta X_n, \quad \text{with } X(0) = Y_n.$$

At each time step t_n we compute the error $\|Y_n - X(t_n)\|$ between the exact and numerical solution and plot these errors over time.

The predicted property for the rank-adaptive and the fixed-rank BUG integrator from theorem 4.1 is verified numerically in figure 6.2. Both integration schemes can reproduce the family of tree tensor networks $X(t)$ exactly up to round-off errors. Choosing a smaller time step size h increases the error in the propagation as the round-off errors can accumulate more during longer computations.
On the contrary, the parallel integrator for tree tensor networks does not share the

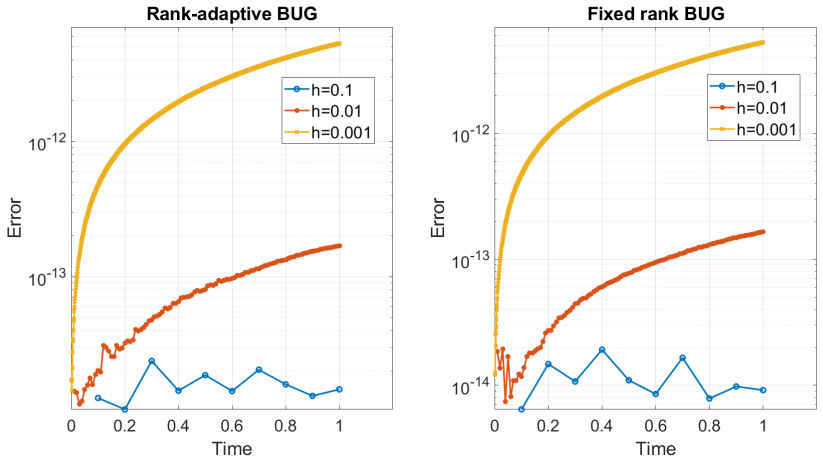

FIGURE 6.2: Left: Error propagation of the rank-adaptive BUG integrator. Right: Error propagation of the fixed-rank BUG integrator.

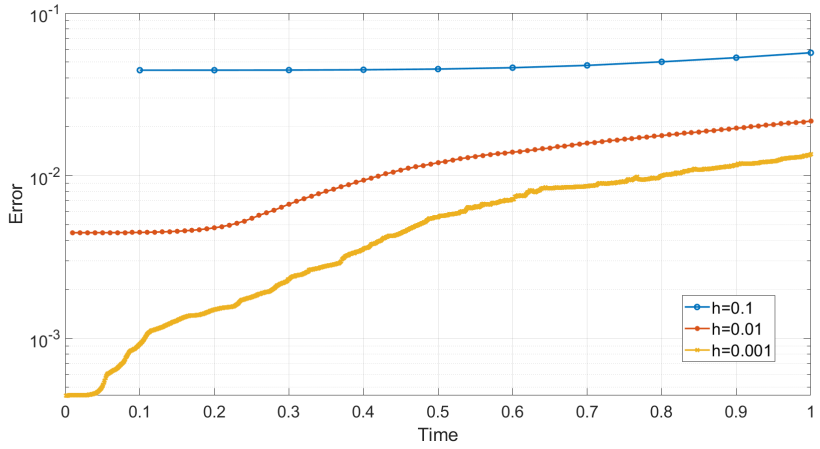

FIGURE 6.3: Error propagation of the parallel BUG integrator.

exactness property. In figure 6.3 we see that the errors in the propagation are of the order of the chosen time step size.

6.2 Closed quantum spin systems

In this section we want to consider the time integration of closed quantum spin systems. The evolution of such systems is described by the Schrödinger equation

$$i\partial_t u = Hu, \quad u(t_0) = u_0, \tag{6.1}$$

where H is a selfadjoint operator. For d distinguishable spin $\frac{1}{2}$-particles, the full state $u(t)$ is an element of the Hilbert space $\mathbb{C}^{2^d}$. This exponential scaling in the Hilbert space size makes direct computations with the full state impossible for a large number

of particles. Therefore, we will represent the state $u(t)$ and also the Hamiltonian H in a tree tensor network format and apply the integrators from chapter 4 and 5.

6.2.1 Ising model in a transverse field

We consider the Ising model in a transverse field with next neighbor interaction, see [Sti73] for more details. The Hamiltonian has the form

$$H = -\Omega \sum_{k=1}^{d} \sigma_x^{(k)} - \sum_{k=1}^{d-1} \sigma_z^{(k)} \sigma_z^{(k+1)}, \tag{6.2}$$

where $\Omega \geq 0$, σ_x and σ_z are the first and third Pauli matrices, respectively, and $\sigma^{(k)} = (\mathbf{I} \otimes \cdots \otimes \mathbf{I} \otimes \sigma \otimes \mathbf{I} \otimes \cdots \otimes \mathbf{I})$ denotes the action of σ on the kth particle. This Hamiltonian encodes an Ising model with next-neighbor interactions, i.e. only neighboring particles interact with each other.

For the following simulations we use as an initial state

$$u_0 = \bigotimes_{k=1}^{d} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \in \mathbb{C}^{2^d}, \tag{6.3}$$

i.e. the state where all particles are spin up. Note that this initial state can be represented by a tree tensor network of rank $r_\tau = 1$ for all $\tau \leq \bar{\tau}$. The basis matrices equal $\mathbf{U}_i = (1, 0)^\top$ for all $i = 1, \ldots, d$ and $C_\tau = 1$ for all core tensors. Although this is theoretically true, it is recommendable to artificially use an initial state of higher rank by using the 2×2 identity matrix as leaves and filling up the core tensors with zeros accordingly. In numerical experiments, it proved to give more accurate solutions.

Error bound

We first want to verify the theoretical error bounds for the rank-adaptive BUG (theorem 4.2), fixed-rank BUG (theorem 4.3) and the parallel BUG (theorem 5.2) by applying these integrators to the Schrödinger equation (6.1) with the Hamiltonian from (6.2). We compare the numerical solution to an exact reference solution obtained from an exact diagonalization. Clearly the solution of (6.1) equals $u(t) = e^{-itH} u_0$. For a system size of $d \leq 8$, it is still feasible to compute the matrix exponential of the Hamiltonian. We then compute the difference in Frobenius norm of the state at time $T = 1$, i.e.

$$\text{error}^{\text{BUG}} = \left\| e^{-iH} u_0 - u^{\text{BUG}} \right\|,$$

where u^{BUG} is the numerical solution obtained by the rank-adaptive, fixed-rank or parallel BUG integrator for tree tensor networks at time $T = 1$.

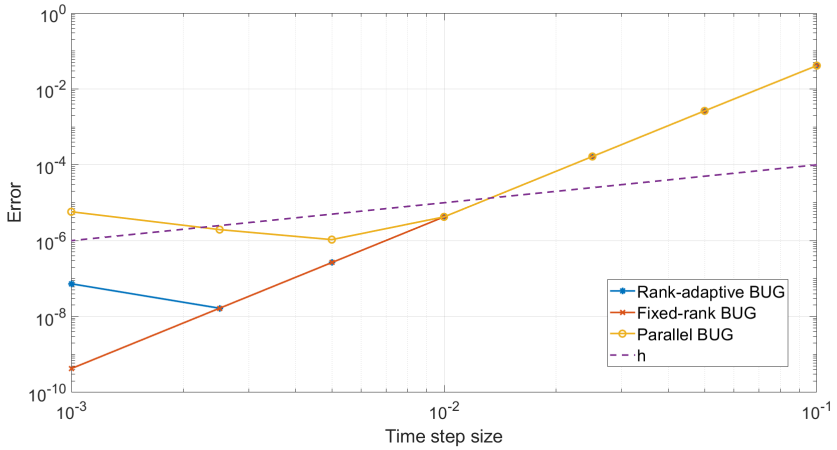

FIGURE 6.4: Error in Frobenius norm at time $T = 1$ for $d = 8$ particles over different time step sizes h for rank-adaptive BUG, fixed-rank BUG and parallel BUG for TTNs. Unspecified parameters are $\Omega = 1$ and $\vartheta = 10^{-14}$.

In figure 6.4 we observe that all three integrators fulfil the theoretical error bound of $\mathcal{O}(h)$. It was already numerically observed in [CEKL24] that the rank-adaptive BUG has a higher order when applied to a non-stiff Schrödinger equation, as is the case here. We further note that the example has rather small size since for $d = 8$ the integrators can use the full Hilbert space to approximate the solution if needed, i.e. no projection error is made.

Norm and energy preservation

By theorem 4.4 and theorem 4.5 we know that the rank-adaptive BUG for TTNs preserves the norm and energy for Schrödinger systems. On the other hand, the fixed-rank BUG and the parallel BUG do not share this property. In figure 6.5 and figure 6.6 we verify these properties. The fixed-rank BUG and the parallel BUG can preserve norm and energy for shorter times but struggle with the preservation for longer times. The rank-adaptive BUG fulfils the theoretical expectations and preserves norm and energy up to the truncation tolerance. Note that by setting the maximal rank $r_{\max} = 30$ for this simulation, the actual truncation tolerance might become larger (and unknown) than the predefined $\vartheta = 10^{-8}$.

Ranks over time

By construction the rank-adaptive BUG and the parallel BUG are rank-adaptive integrators. Given a fixed tolerance ϑ, it is of interest how the ranks evolve for both methods. In figure 6.7 we observe that the the rank-adaptive BUG integrator chooses smaller maximal tree ranks than the the parallel BUG integrator. A similar behaviour is also observed for different parameter choices.

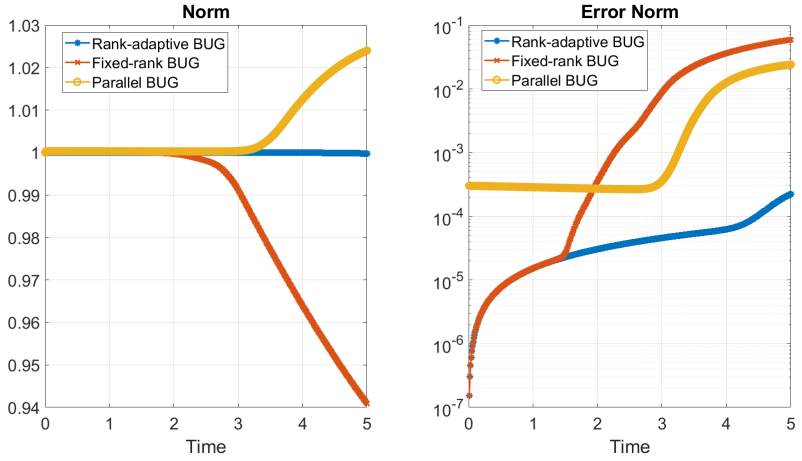

FIGURE 6.5: Norm (left) and error of the norm (right) over time for the rank-adaptive BUG, fixed-rank BUG and parallel BUG for TTNs. Unspecified parameters are $d = 16$, $\Omega = 1$, $\vartheta = 10^{-8}$, $h = 0.01$ and $r_{\max} = 30$.

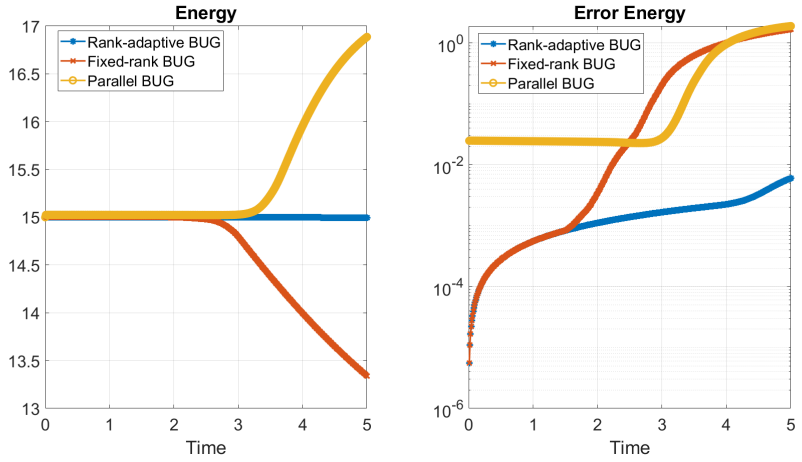

FIGURE 6.6: Energy (left) and error of the energy (right) over time for the rank-adaptive BUG, fixed-rank BUG and parallel BUG for TTNs. Unspecified parameters are $d = 16$, $\Omega = 1$, $\vartheta = 10^{-8}$, $h = 0.01$ and $r_{\max} = 30$.

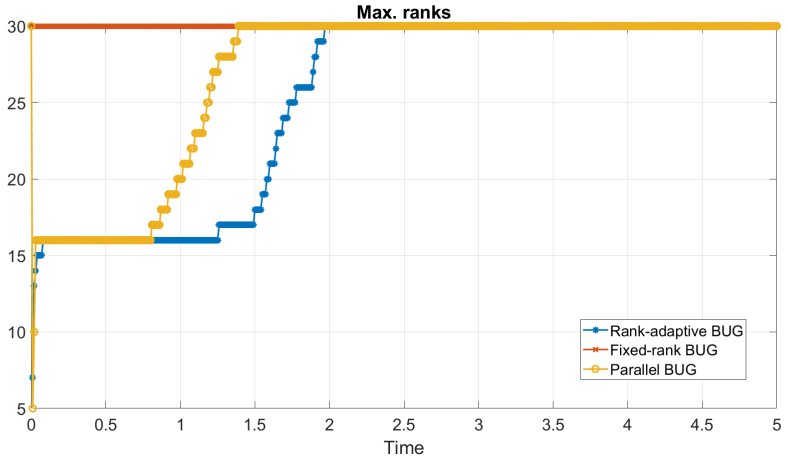

FIGURE 6.7: Maximal tree rank over time for the rank-adaptive BUG, fixed-rank BUG and the parallel BUG for TTNs. Unspecified parameters are $d = 16$, $\Omega = 1$, $\vartheta = 10^{-8}$, $h = 0.01$ and $r_{\max} = 30$.

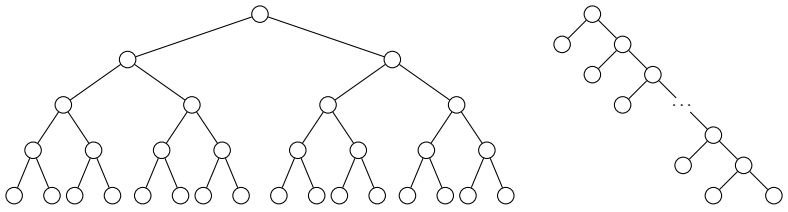

FIGURE 6.8: Left: Balanced binary tree. Right: Tensor train/matrix product state.

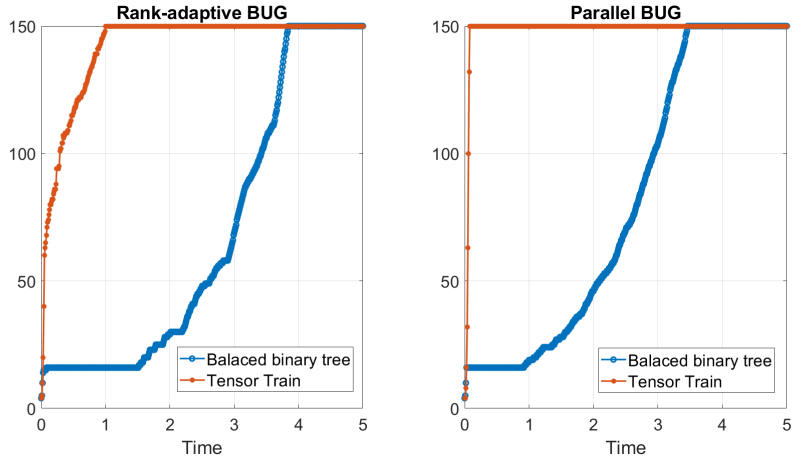

FIGURE 6.9: Maximal tree rank over time for the rank-adaptive BUG (left) and the parallel BUG (right) for a balanced binary tree and a tensor train. Unspecified parameters are $d = 16$, $\Omega = 1$, $\vartheta = 10^{-8}$, $h = 0.01$ and $r_{\max} = 150$.

Comparison of different tree structures

Since each tensor can be approximated in different tree formats, it is interesting to investigate the tree structure's influence on the simulation. For comparison, we choose balanced binary trees and the tensor train format. Both are standard ansatz for quantum simulations. Graphically the tree structures are illustrated in figure 6.8 for $d = 16$ particles, which was taken from [CLS23, Figure 7.3]. In figure 6.9 the maximal tree rank after each time step with the rank-adaptive and the parallel BUG is plotted for both tree formats, allowing for a maximal rank of $r_{\max} = 150$. We see that for both integrators the tensor train format requires much higher ranks compared to the balanced binary tree. Even though the Hamiltonian only encodes next-neighbor interactions, the tensor train format struggles to capture long-range effects. However, the maximal rank is only one measure of efficiency. The total memory requirement to store the state at each time step is of interest for two reasons. First, when dealing with high-dimensional tensor differential equations, memory can become scarce. Second, the memory requirement is correlated with the overall computational complexity of the simulation. In figure 6.10 we observe that the memory requirements behave similarly to the maximal ranks. The balanced binary tree can capture the dynamics for longer times with very little memory, independent of the chosen integrator. Note that the parallel BUG chooses again mildly higher ranks than the rank-adaptive BUG, which was already observed in figure 6.7.

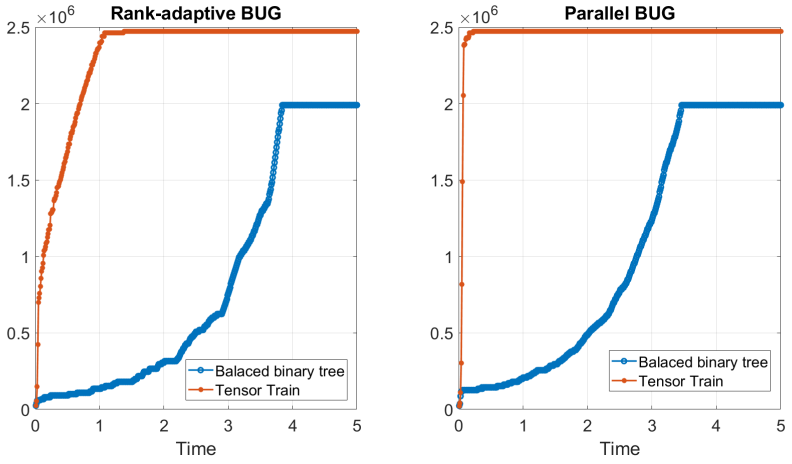

FIGURE 6.10: Memory requirements in bytes to store the approximation at each time step for the rank-adaptive BUG (left) and the parallel BUG (right) for a balanced binary tree and a tensor train. Unspecified parameters are $d = 16$, $\Omega = 1$, $\vartheta = 10^{-8}$, $h = 0.01$ and $r_{\max} = 150$.

6.2.2 Ising model with long-range interactions

Recall the long-range Hamiltonian for unitary dynamics from chapter 3, with which we want so solve the Schrödinger equation, i.e.

$$i\partial_t \psi = H\psi, \quad \text{with } H = \Omega \sum_{k=1}^{d} \boldsymbol{\sigma}_x^{(k)} + \Delta \sum_{k=1}^{d} \boldsymbol{n}^{(k)} + \nu \sum_{k \neq h}^{d} \frac{1}{|k-h|^{\alpha}} \boldsymbol{n}^{(k)} \boldsymbol{n}^{(h)}. \tag{6.4}$$

The long-range interactions make the corresponding TTNO and the time integration far more complicated in terms of memory requirement and computational time. Nevertheless, the class of BUG integrators is a suitable method for computing the dynamics of the state. Again, for all following simulations, we use the state where all particles are spin up as an initial state, c.f. (6.3).

Error bound

As for the previous model, we want to verify the error bounds for the rank-adaptive, fixed-rank and parallel BUG integrator. We compare the approximated solutions of those to the numerically exact solution again obtained by an exact diagonalization, see subsection 6.2.1 for details. Similar to the next neighbor example from subsection 6.2.1, we see that the parallel BUG gives a coarser approximation than the rank-adaptive BUG and the fixed-rank BUG. Again, the latter two integrators give better approximations than the error bounds of $\mathcal{O}(h)$ from theorem 4.2 and theorem 4.3.

Time integration using approximated Hamiltonians

In subsection 3.3.3 of chapter 3 we presented a way to construct ϵ-approximations of

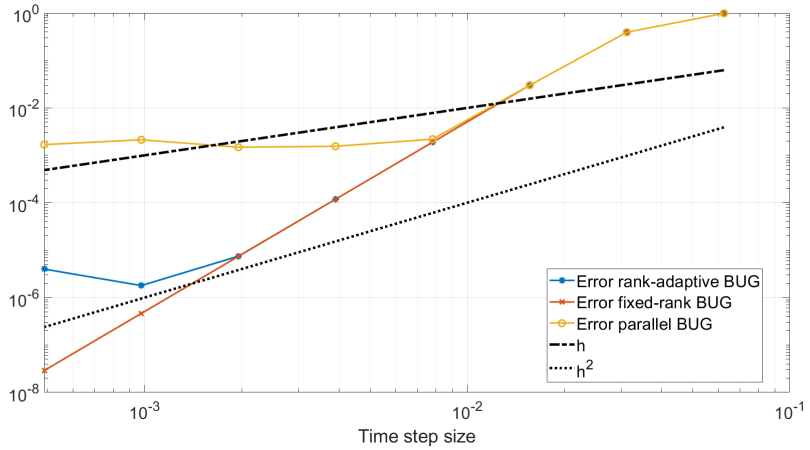

FIGURE 6.11: Error in Frobenius norm at time $T = 1$ for $d = 8$ particles over different time step sizes h for rank-adaptive BUG, fixed-rank BUG and parallel BUG for TTNs. Unspecified parameters are $\Omega = 1$, $\Delta = 1$, $\nu = 2$ and $\vartheta = 10^{-8}$.

Hamiltonian using the HSS decomposition. We already verified the error bounds of the approximated TTNO. We now want to investigate how an approximated Hamiltonian influences the time integration.

For this we compute the dynamics of the Schrödinger equation (6.4) using the rank-adaptive, fixed-rank and parallel BUG together with approximated Hamiltonians for different HSS tolerances ϵ. At each time step, we compute the distance between the BUG solution using the approximated and the exact TTNO representation, i.e.

$$\|X_{\mathrm{BUG}}(t) - X^{\epsilon}_{\mathrm{BUG}}(t)\|,$$

where $X_{\mathrm{BUG}}(t)$ is the BUG solution using the exact TTNO representation and $X^{\epsilon}_{\mathrm{BUG}}(t)$ the BUG solution using the ϵ-approximated TTNO representation. The exact TTNO representation of the Hamiltonian is obtained by applying the HSS construction with tolerance $\epsilon = 10^{-16}$.

In figure 6.12 we plot the difference $\|X_{\mathrm{BUG}}(t) - X^{\epsilon}_{\mathrm{BUG}}(t)\|$ over time. We observe that the fixed-rank and parallel BUG are more sensitive when using an approximated TTNO for computing the dynamics. Even for a small HSS tolerance of $\epsilon = 10^{-10}$ the difference is only mildly reduced to 10^{-5} for the fixed-rank and 10^{-2} for the parallel BUG. In contrast, the difference of the solutions using the rank-adaptive BUG is of the same order as the chosen tolerance ϵ. Thus, the rank-adaptive BUG appears to be a suitable method when working with approximated Hamiltonians. The same behaviour is observed when computing the difference in the magnetization in z-direction, see figure 6.13. Note that the curves for $\epsilon = 10^{-12}$ are not visible as they are numerically already zero.

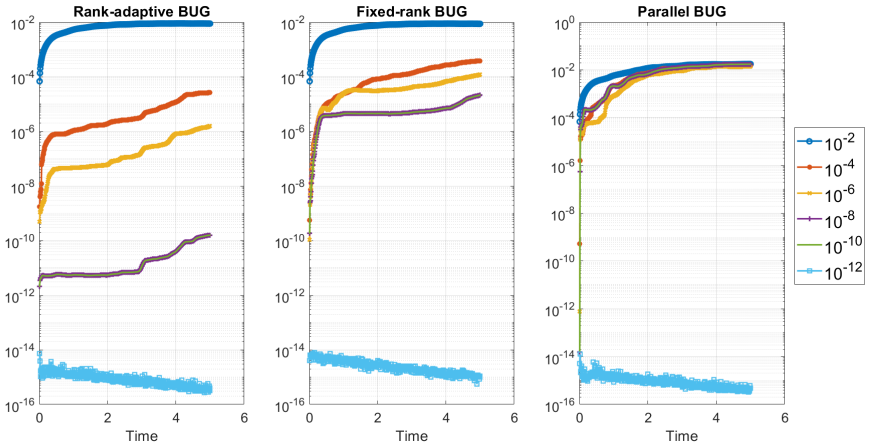

FIGURE 6.12: Difference in Frobenius norm between the BUG solutions using the exact and the ϵ-approximated TTNO representation of the Hamiltonian for the rank-adaptive BUG, fixed-rank BUG and the parallel BUG. Unspecified parameters are $d = 16$, $\Omega = 1$, $\Delta = 1$, $\nu = 2$, $\vartheta = 10^{-8}$, $h = 0.01$ and $r_{\max} = 30$.

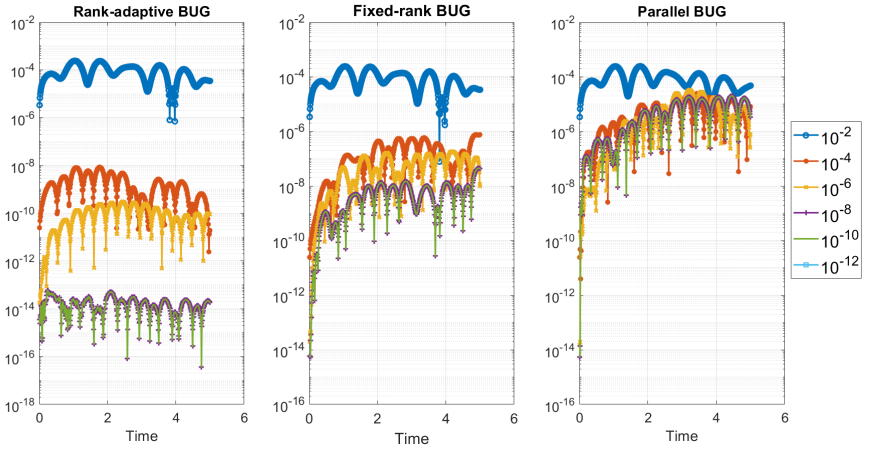

FIGURE 6.13: Difference between the magnetization in z-direction of the BUG solutions using the exact and the ϵ-approximated TTNO representation of the Hamiltonian for the rank-adaptive BUG, fixed-rank BUG and the parallel BUG. Unspecified parameters are $d = 16$, $\Omega = 1$, $\Delta = 1$, $\nu = 2$, $\vartheta = 10^{-8}$, $h = 0.01$ and $r_{\max} = 30$.

Comparison of different tree structures

As in the previous subsection, we compare the influence of different tree structures. We do this for the long-range Hamiltonian (6.4) and check how introducing long-range interactions in the Hamiltonian influences the maximal tree ranks and memory requirements. We choose the balanced binary tree and the tensor train format, illustrated already in figure 6.8.

Figure 6.14 investigates the maximal ranks, while figure 6.15 investigates the memory footprint. Similar to the next-neighbor example from the previous subsection, the TTN based on a balanced binary tree can capture the dynamics with less rank and a strongly reduced memory footprint than the tensor train format. Interestingly, the parallel BUG

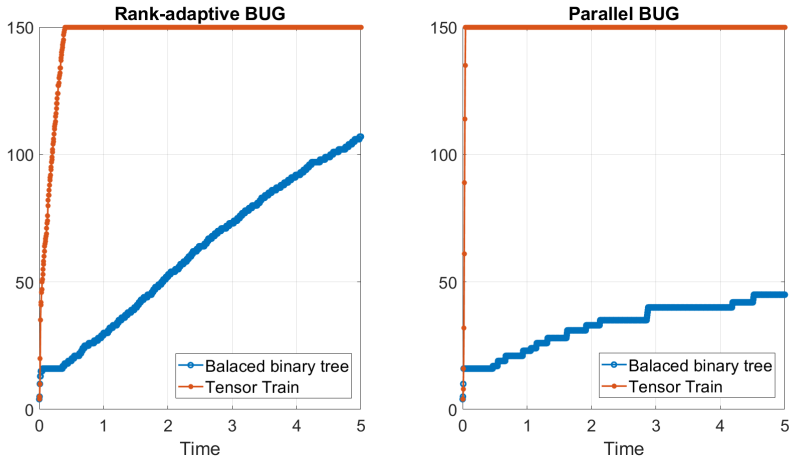

FIGURE 6.14: Maximal tree rank over time for the rank-adaptive BUG (left) and the parallel BUG (right) for a balanced binary tree and a tensor train. Unspecified parameters are $d = 16$, $\Omega = 1$, $\Delta = 1$, $\nu = 2$, $\alpha = 1$, $\vartheta = 10^{-8}$, $h = 0.005$ and $r_{\max} = 150$.

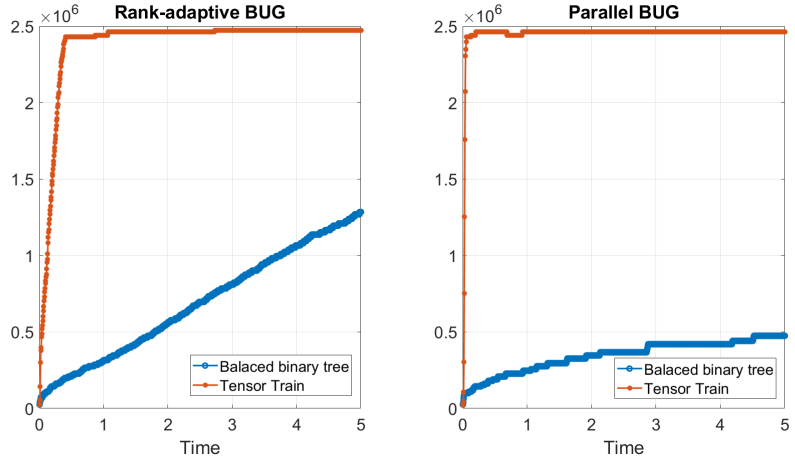

FIGURE 6.15: Memory requirements in bytes to store the approximation at each time step for the rank-adaptive BUG (left) and the parallel BUG (right) for a balanced binary tree and a tensor train. Unspecified parameters are $d = 16$, $\Omega = 1$, $\Delta = 1$, $\nu = 2$, $\alpha = 1$, $\vartheta = 10^{-8}$, $h = 0.005$ and $r_{\max} = 150$.

integrator chooses smaller ranks compared to the rank-adaptive BUG, which was different for the example with only next-neighbor interactions. We conclude that the maximal ranks over time are strongly influenced by the chosen time-integration method and the underlying model.

We know from chapter 3 that the TTNO representation in the balanced binary tree format is more compact than the tensor train format (cf. figure 3.8). Since by figure 6.14 and figure 6.15 we observe that the balanced binary tree can capture the dynamics with smaller ranks and less complexity, the balanced binary tree format seems to be better suited for simulations with quantum spin systems. The balanced binary tree appears to encode the long-range correlations in a more compact low-rank structure than the tensor train format.

6.3 Open quantum spin systems

In this chapter we will discuss the application of the presented integrators to a one-dimensional long-range dissipative Ising model. The results of this section are based on the work "Numerical simulations of long-range open quantum many-body dynamics with tree tensor networks" by Christian Lubich, Gianluca Ceruti, Igor Lesanovsky, Federico Carollo and the author [SLC+24].
Recall the quantum master equation (3.27) from chapter 3, where the (vectorized) density matrix $\rho(t)$ evolves through

$$\dot{\rho}(t) = \mathcal{L}[\rho(t)] := -i[H, \rho(t)] + \mathcal{D}[\rho(t)],$$

and the Hamiltonian H and the decay operator $\mathcal{D}$ are defined as

$$H = \Omega \sum_{k=1}^{D} \boldsymbol{\sigma}_x^{(k)} + \Delta \sum_{k=1}^{D} \boldsymbol{n}^{(k)} + \frac{\nu}{2c_\alpha} \sum_{k \neq h}^{D} \frac{\boldsymbol{n}^{(k)} \boldsymbol{n}^{(h)}}{|k-h|^\alpha}$$
$$\mathcal{D} = \sum_{k=1}^{D} \left[\boldsymbol{J} \otimes (\boldsymbol{J}^*)^\top - \frac{1}{2} \boldsymbol{J}^* \boldsymbol{J} \otimes \mathbf{I} - \frac{1}{2} \mathbf{I} \otimes (\boldsymbol{J}^* \boldsymbol{J})^\top \right]^{(k)},$$

see chapter 3 for details. Note that the decay rate γ is hidden in the matrix $J = \sqrt{\gamma} \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$. Due to the decay operator, we have a dissipative and thus time-irreversible system. For this reason one should not apply the projector-splitting integrator [CLW21], as the **S**-step/core update is solved backwards in time. The proposed BUG integrators in the present thesis do not have this backward step in time and are therefore better suited for numerical simulations of this dissipative model.

In [SLC+24], the author, together with Christian Lubich, Gianluca Ceruti, Igor Lesanovsky and Federico Carollo, investigated the model for phase transitions concerning the ratio $\frac{\Omega}{\gamma}$ against the stationary state density $< n >$. $\frac{\Omega}{\gamma}$ encodes the ratio between the strength of driving particles into an excited state and the dissipation of the system. The observable $\langle n \rangle$ is defined by

$$\langle n \rangle = \frac{1}{D} \langle \psi^{\text{flat}} | \sum_{k=1}^{D} (\boldsymbol{n} \otimes \mathbf{I}_2)^{(k)} | \rho(t) \rangle,$$

where $\rho(t)$ denotes the solution of (3.27), $\mathbf{I}_2$ the 2×2 identity matrix and $\psi^{\text{flat}} = \otimes_{k=1}^{D} \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}^\top$ the vector representation of the identity. For all simulations, the fixed-rank BUG integrator for TTNs was used, since this showed the most reduced computational cost. The stationary state density is obtained by computing the dynamics of the model for long times (here $T = 15$) and then applying the observable to the stationary state.

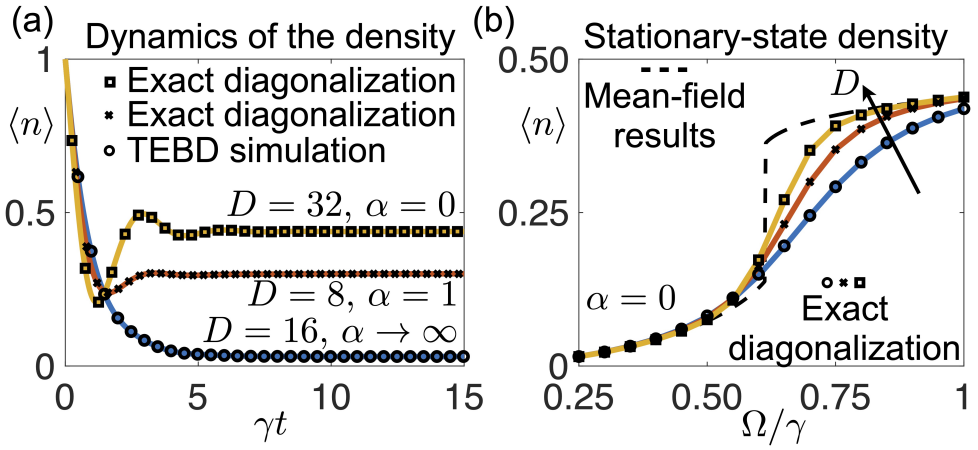

FIGURE 6.16: Left: Density $\langle n \rangle$ over time computed with the fixed-rank BUG integrator (solid lines) and other approaches for different regimes of α. Right: Numerical convergence to the mean-field limit for $\alpha = 0$. The unspecified parameters for both plots are $\Delta = -2$ and $\nu = 5$.

First, we check if the integrator can capture the physics of this model. Therefore, we compute reference solutions for all interesting regimes, i.e. all-to-all interactions ($\alpha = 0$) for $D = 32$, long-range interactions ($\alpha = 1$) for $D = 8$ and next-neighbor interactions ($\alpha = \infty$) for $D = 16$. The reference solutions for the first two cases are obtained by an exact diagonalization of the Lindblad generator. The reference solution for the next-neighbor case is obtained by a matrix product state simulation with a time-evolving- block-decimation (TEBD) algorithm, see [PKS+19] for a detailed description of the TEBD method. In figure 6.16 (a) we see that the fixed-rank BUG integrator for TTNs agrees with the reference solution in all interaction regimes. Hence, the integrator allows us now to interpolate between the different regimes. With this knowledge, we aim now to investigate the phase transitions. Note that figure 6.16 was taken from [SLC+24, Figure 3].

In the case of all-to-all interactions of the sites, i.e. $\alpha = 0$, it is known that the mean-field limit (i.e. $D \to \infty$) indeed shows a phase transition [BCFN18]. In figure 6.16 (b) we show the density $\langle n \rangle$ at a stationary point for different ratios of $\frac{\Omega}{\gamma}$, for $D = 8, 16, 32$ and maximal ranks $r_{\max} = 10, 20, 30$ respectively. The dashed line shows the mean-field prediction, which is exact in the limit $D \to \infty$ [BCFN18]. We observe that the fixed-rank BUG integrator shows a numerical convergence of the stationary state density to the mean-field limit with respect to the number of particles, which shows a phase transition. On the other hand, it is known that for $\alpha = \infty$ the phase transition is not present anymore [JBV+18]. Hence, the question arises for which values $0 < \alpha < \infty$ the phase transition persists.

For each value of α, we compute the stationary density for different ratios $\frac{\Omega}{\gamma}$. Thus, for each value of α, we obtain a curve of $\frac{\Omega}{\gamma}$ vs. $\langle n \rangle$ at final time T. In figure 6.17 we see

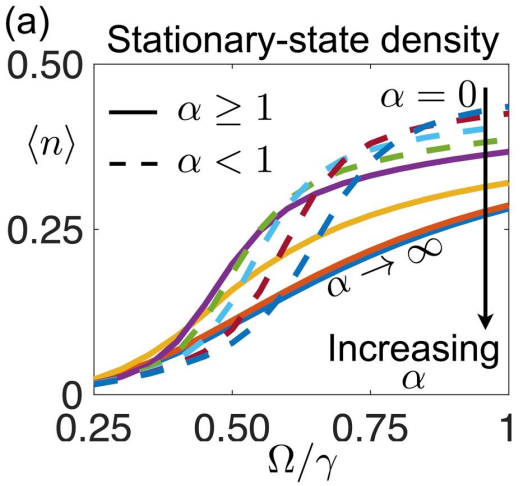

FIGURE 6.17: Stationary behaviour of the density $\langle n \rangle$ for $D = 16$ particles as a function of $\frac{\Omega}{\gamma}$ for $\alpha = 0, 0.25, 0.50, 0.75, 1, 2, 5$ and $\alpha = \infty$. The unspecified parameters are $\Delta = -2$, $\nu = 5$, $\gamma T = 15$ and $r_{\max} = 30$ (note for $\alpha = 0$, $r_{\max} = 20$ was already enough).

that for $\alpha > 1$ the density $\langle n \rangle$ behaves similarly to the curve of $\alpha = \infty$, where $\langle n \rangle$ seems to be smooth as a function of $\frac{\Omega}{\gamma}$ (solid lines). For $\alpha \leq 1$ (dashed lines), there seems to be an emergence of a sharp crossover which looks very similar to the $\alpha = 0$ case. We note that figure 6.17 was taken from [SLC+24, Figure 4]. Hence, the numerical simulations indicate a persistence of the phase transition for all values $\alpha \leq 1$.
These numerical findings have been confirmed by a novel work which theoretically investigates a class of models, including the one used in our simulations here. Interestingly, the authors proved in [MLC24] that the presented numerical indications were indeed right and the phase transition persists for this model for all values $\alpha \leq 1$.

6.4 Henon-Heiles

The time-dependent Schrödinger equation with a Henon-Heiles potential, modelling a coupled oscillator, is a challenging problem from quantum molecular dynamics. The problem writes as

$$i\partial_t \psi = H\psi, \quad \text{with}$$
$$H(x_1, \ldots, x_d) = -\frac{1}{2}\Delta + \frac{1}{2}\sum_{k=1}^{d} x_k^2 + \lambda \sum_{k=1}^{d-1} \left(x_k^2 x_{k+1} - \frac{1}{2} x_{k+1}^3 \right), \tag{6.5}$$

where Δ denotes the Laplace operator and $\lambda = 0.11180$. The first two sums in (6.5) encode the harmonic part, the other terms encode the anharmonic part of the Hamiltonian. We refer to [NM02] for a detailed description of the model.

This problem is fundamentally different from the quantum spin systems we considered before. The Schrödinger equation here is continuous in time and space, while the systems before were only continuous in time. Thus, we first need to discretize in space to represent the state ψ and the Hamiltonian H in tree tensor network format.

We follow [LOV15] for the discretization of the problem. The main difference with [LOV15] lies in the fact that the authors used a tensor train ansatz together with the projector-splitting integrator to approximate the solution. We consider this problem on the d-dimensional cube $[-10, 10]^d$. As (6.5) is defined on the full space $\mathbb{R}^d$, we need appropriate boundary conditions. To deal with that we choose the same complex absorbing potential (CAP) as in [LOV15], which is defined as

$$W(x_1, ..., x_d) = -i \sum_{k=1}^{d} \left((x_k - 6)_+^3 + (x_k + 6)_-^3 \right). \tag{6.6}$$

Here z_+ and z_- are defined as

$$z_+ = \begin{cases} z, & \text{if } z \geq 0 \\ 0, & \text{else} \end{cases} \quad \text{and} \quad z_- = \begin{cases} z, & \text{if } z \leq 0 \\ 0, & \text{else} \end{cases}.$$

For the discretization of the problem we are taking a discrete variable representation (DVR) basis with 31 basis functions in each dimension over the interval $(-10, 10)$. For details on DVR basis representation we refer to the next paragraph. The initial wave function ψ_0 is represented as a binary TTN with 31 basis functions in each degree of freedom, i.e. $n_l = 31$ for all $l \in L(\bar{\tau})$.

As an initial function ψ_0 at time $t = 0$, we choose a product of shifted, one-dimensional Gaussians

$$\psi_0(x) = \prod_{k=1}^{d} \exp\left(-\frac{(x_k - 2)^2}{2} \right).$$

In the following, we present how the problem can be discretized and how one can apply dynamical low-rank robust numerical integrators to the problem.

Discrete Variable Representation and Precomputations

The discrete variable representation is a method to represent a wave function in a primitive basis. Here, we follow the ideas in [BJWM00, Appendix B]. There are many possible choices for the basis, e.g. Legendre polynomials. We consider the basis functions

$$\phi_n(x) = \left(\frac{1}{b-a} \right)^{1/2} \exp\left(\frac{2\pi i n (x-a)}{b-a} \right) \quad \text{for } \forall n \in \mathbb{N}.$$

The family $(\phi_n)_{n\in\mathbb{N}}$ defines an orthonormal L^2-basis and is equivalent to a Fourier basis. With this definition, we easily obtain

$$\frac{d^2}{dx^2}\phi_n(x) = \left(\frac{2i\pi n}{b-a}\right)^2 \phi_n(x).$$

We follow the MCTDH ansatz. For each single particle function, consider a truncated basis set with K_i primitive basis functions $(\phi_n^{(i)})_{n=1}^{K_i}$ and with $x = (x_1, \dots, x_d)$ represent the ψ in the truncated basis by

$$\psi(x,t) = \sum_{j_1=1}^{K_1} \cdots \sum_{j_d=1}^{K_d} A_{j_1,\dots,j_d}(t) \prod_{n=1}^{d} \phi_{j_n}^{(n)}(x_n). \tag{6.7}$$

The tensor $A(t) = (A_{j_1,\dots,j_d}(t)) \in \mathbb{C}^{K_1\times\cdots\times K_d}$ does now only depend on time, while the primitive basis is time-independent. To evolve the coefficients over time we rewrite (6.5) via the Bra-Ket notation and by testing the equation in the primitive basis we obtain

$$\langle \prod_{k=1}^{d} \phi_{j_k}^{(n)} | \dot{\psi}(x,t)\rangle = \langle \prod_{k=1}^{d} \phi_{j_k}^{(n)} | -iH|\psi(x,t)\rangle, \tag{6.8}$$

for all possible combinations of basis functions $\phi_{j_k}^{(n)}$. Inserting (6.7) for ψ and using the orthonormality of the primitive DVR basis we obtain the ODE

$$\sum_{j_1,\dots,j_d=1} \dot{A}_{j_1,\dots,j_d}(t) = \sum_{j_1,\dots,j_d=1} A_{j_1,\dots,j_d}(t)\ \langle \prod_{n=1}^{d} \phi_{j_n}^{(n)} | -iH| \prod_{n=1}^{d} \phi_{j_n}^{(n)}\rangle. \tag{6.9}$$

The remaining inner products in (6.9) can be all precomputed since they do not depend on time. Due to that all $\phi_i^{(n)}$ are one-dimensional functions, the large inner product $\langle \prod_{n=1}^d \phi_{j_n}^{(n)} | -iH| \prod_{n=1}^d \phi_{j_n}^{(n)}\rangle$ on $\mathbb{R}^d$ boils down to compute d inner products in one dimension. Those one-dimensional integrals are approximated with a high-order quadrature formula.

For the Laplace part for example, the integral is of the form

$$\frac{-i}{2}\langle \phi_{j_i}^{(n)} | \frac{d^2}{dx^2} | \phi_{j_k}^{(n)}\rangle.$$

For $i = 1, \dots, d$ define the matrix $\boldsymbol{D}^i = (\frac{-i}{2}\langle \phi_j^{(i)} | \frac{d^2}{dx^2} | \phi_k^{(i)}\rangle)_{j,k=1}^{K_i}$. Analogously, one can precompute all integrals for the potential. Denote by $\boldsymbol{M}_1^i$ the matrix for the inner products corresponding to x_i^2, by $\boldsymbol{M}_2^i$ the one for x_i, by $\boldsymbol{M}_3^i$ the one for x_i^3 and by $\boldsymbol{W}^i$ the one for the CAP potential. Altogether, this gives us a tensor ODE of the form

$$\dot{A}(t) = A(t)\left(\bigtimes_{i=1}^{d}\left(\boldsymbol{D}^{(i)} + \boldsymbol{M}_1^{(i)} + \boldsymbol{M}_3^{(i)} + \boldsymbol{W}^{(i)}\right) + \bigtimes_{i=1}^{d-1} \boldsymbol{M}_1^{(i)}\boldsymbol{M}_2^{(i+1)}\right), \tag{6.10}$$

where $\boldsymbol{D}^{(i)}$ denotes the action of the matrix $\boldsymbol{D}^i$ on the ith site and analogously for the other matrices. Further, we introduce the notation

$$TA(t) \coloneqq A(t) \bigtimes_{i=1}^{d} \boldsymbol{D}^{(i)} \tag{6.11}$$

$$VA(t) \coloneqq A(t) \left(\bigtimes_{i=1}^{d} \left(\boldsymbol{M}_1^{(i)} + \boldsymbol{M}_3^{(i)} + \boldsymbol{W}^{(i)} \right) + \bigtimes_{i=1}^{d-1} \boldsymbol{M}_1^{(i)} \boldsymbol{M}_2^{(i+1)} \right). \tag{6.12}$$

Finally, we represent $A(t)$ in a tree tensor network format, which encodes now all coefficients of the DVR basis representation, and apply the BUG integrators to solve the tensor ODE (6.10). Note that the TTNO corresponding to the operator in (6.10) can be represented as a rank four TTNO, as only next-neighbor coupling is enabled, cf. the discussion in chapter 3.

Splitting

Due to the Laplacian part in (6.5) and (6.10) the problem is stiff. If an explicit time integration is performed in the proposed BUG integrators, this will require very small time-step sizes. Indeed, if the BUG integrators are applied with a moderate time step size (say $h = 0.01$) the numerical solution is observed to become often unstable.
To overcome this issue, we apply a standard Strang splitting for the discretized Schrödinger equation (6.10).

$$e^{-ihH} \approx e^{-i\frac{h}{2}V} e^{-ihT} e^{-i\frac{h}{2}V}, \tag{6.13}$$

i.e. we perform a half-step only with the potential followed by a full step with the kinetic part and the second half-step with the potential. The Strang splitting is a second-order approximation in time and one of the most used splitting schemes in computations [Lub08]. By the splitting, the stiff part T and the non-stiff part V can be treated separately. The free Schrödinger equation with only the discretized Laplace operator can be written (cf. subsection 3.1.1) as

$$\dot{u}(t) = Tu(t) = u(t) \bigtimes_{j=1}^{d} \boldsymbol{D}^j, \quad u(0) = u_0,$$

where $\boldsymbol{D}^j$ is a discretization of the second derivative in jth direction. Notably, this part can be treated exactly. The exact solution for the discretized free Schrödinger equation can be written as

$$u(t) = u_0 \bigtimes_{j=1}^{d} e^{-it\boldsymbol{D}^j},$$

which is clear by taking the time derivative of $u(t)$. To approximate the Schrödinger equation having only the potential V, we apply the BUG integrators from chapter 4 and chapter 5.

Numerical results

In quantum chemistry, the vibrational spectrum of a molecule is of interest and is obtained as follows. Denote by $X(t)$ the numerical solution at time t. At each time step, we compute the autocorrelation function defined by $a(t) = \langle X(t), X(0) \rangle$, i.e. the overlap of $X(t)$ with the initial data. As in [NM02], the spectrum is now calculated via the Fourier transform of the autocorrelation function $a(t)$ by

$$S(\omega) = \frac{1}{\pi} \mathrm{Re} \int_0^\infty e^{-i\omega t} a(t) \, dt.$$

If the dynamic is approximated accurately, $S(\omega)$ will be a sum of delta functions, where each peak is located at an eigenvalue of the operator H. For a sufficiently accurate solution one must compute the dynamics for long times.

In numerical simulations for the Henon-Heiles potential, the parallel BUG Integrator for tree tensor networks appeared to be numerically more unstable. Already for the matrix case (i.e. $d = 2$), norm and energy behave very nonphysical and the vibrational spectrum does not show the expected behaviour, whereas the rank-adaptive and fixed-rank BUG produce the expected solutions, see figure 6.18 for the vibrational spectrum and figure 6.19 for norm and energy conservation. The rank-adaptive BUG preserves the norm and the energy better than the fixed-rank BUG, which verifies our theoretical results from chapter 4 again. Note that decreasing the time step size improves the results of the parallel BUG integrator but one needs significantly smaller time step sizes to obtain similar results to the rank-adaptive or fixed-rank BUG. Increasing the rank did not affect the simulation. Therefore, we only consider the rank-adaptive and the fixed-rank BUG in the following simulations.

We now turn to simulations with a larger number of particles. In figure 6.20 we observe that the fixed-rank BUG produces the expected results (cf. [NM02]), i.e. peaks at the eigenvalues of the Hamiltonian on top of a Gaussian function. Surprisingly, the rank-adaptive BUG does not produce these results but performs similarly to the parallel BUG in figure 6.18. Increasing the ranks did not improve the vibrational spectrum. We investigate this in more detail. In figure 6.21 we plot the autcorrelation function over time, computed with both integrators. We observe that the rank-adaptive BUG fails to capture the dynamics. After short times, the autocorrelation behaves as for the fixed-rank BUG but for longer times the rank-adaptive BUG fails to produce the smaller peaks in the autocorrelation. Therefore, the rank-adaptive BUG is only capable of producing a

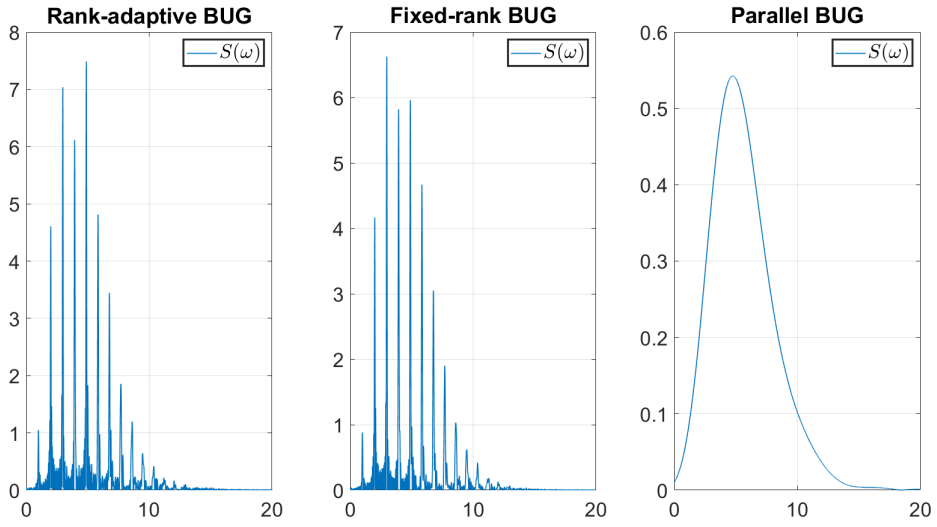

FIGURE 6.18: Vibrational spectrum for $d = 2$ particles computed with the rank-adaptive BUG (left), fixed-rank BUG (middle) and parallel BUG (right). Unspecified parameters are $T = 60$, $h = 0.01$, $r_{\max} = 4$.

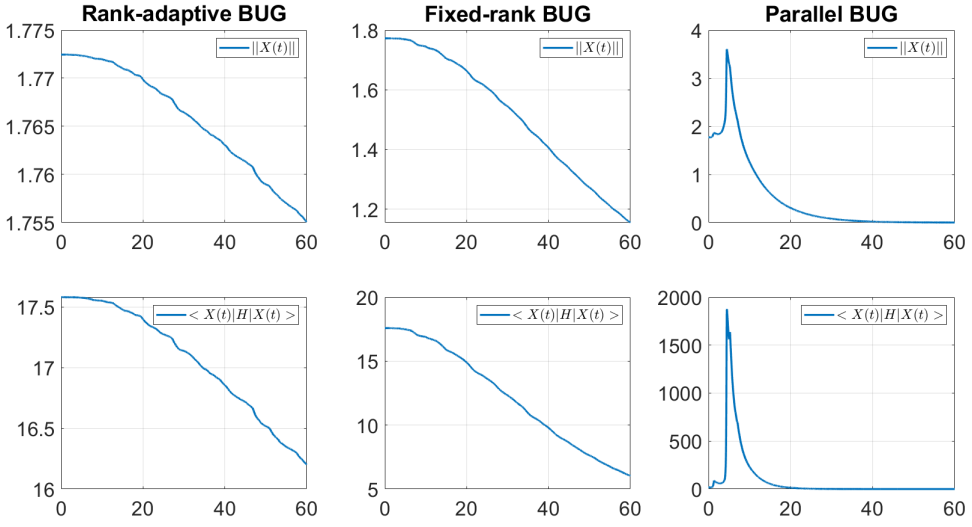

FIGURE 6.19: Norm (top) and energy (bottom) over time for $d = 2$ particles computed with the rank-adaptive BUG (left), fixed-rank BUG (middle) and parallel BUG (right). Unspecified parameters are $T = 60$, $h = 0.01$, $r_{\max} = 4$.

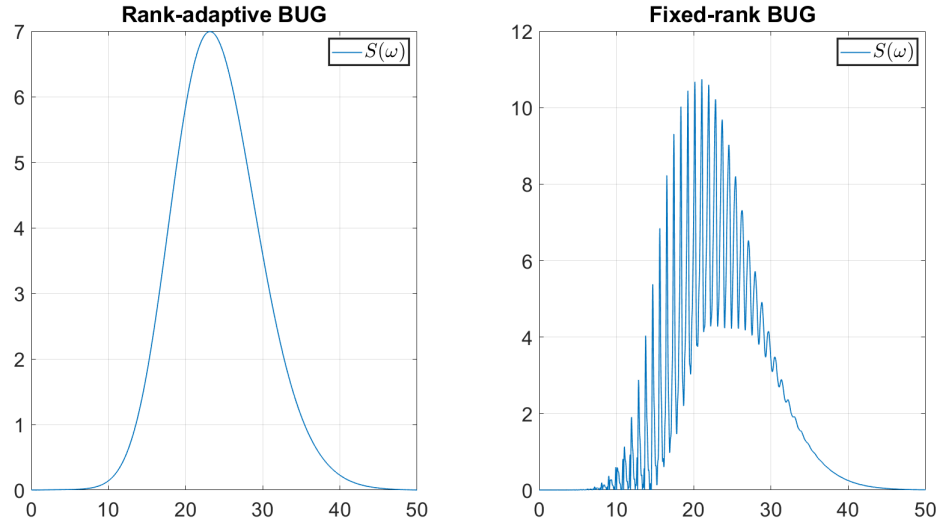

FIGURE 6.20: Vibrational spectrum for $d = 8$ particles computed with the rank-adaptive BUG (left) and fixed-rank BUG (right). Unspecified parameters are $T = 30$, $h = 0.005$, $r_{\max} = 4$.

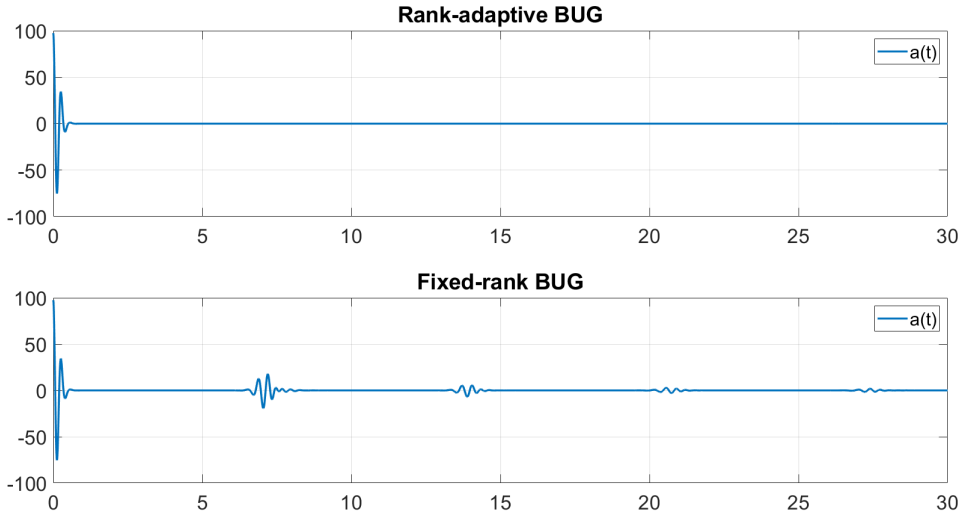

FIGURE 6.21: Autocorrelation function $a(t)$ for $d = 8$ particles computed with the rank-adaptive BUG (top) and fixed-rank BUG (bottom). Unspecified parameters are $T = 30$, $h = 0.005$, $r_{\max} = 4$.

Gaussian function without the peaks. Similar results are obtained for higher-dimensional simulations, see figure 6.22. Only the fixed-rank BUG produced the expected vibrational spectra. This should be investigated in more detail in the future. One potential issue might arise from the truncation procedure where only high-energy basis functions are preserved along the time integration. Finally, note that all simulations in figure 6.22 showed good approximations when using a maximal rank of four and six, respectively. Thus, the problem appears to be very much of a low-rank structure.

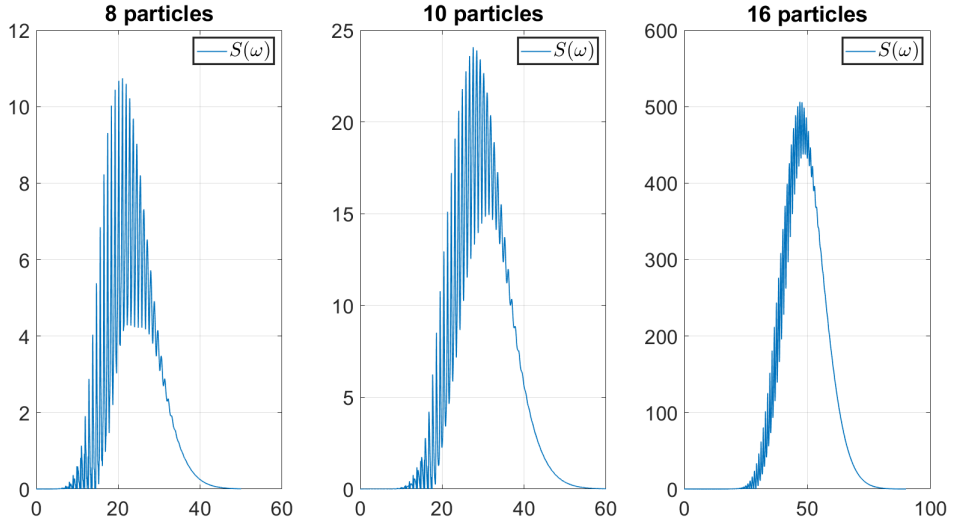

FIGURE 6.22: Vibrational spectrum for $d = 8$ (left), $d = 10$ (middle) and $d = 16$ (right) particles computed with the fixed-rank BUG (right). Unspecified parameters are: for all simulations $h = 0.005$. Further, left $T = 30$ and $r_{\max} = 4$, middle $T = 60$ and $r_{\max} = 4$, right $T = 50$ and $r_{\max} = 6$.

Chapter 7

Conclusion

To summarise the results from chapter 6, we see that the choice of an integrator is very much problem-dependent. There is no clear superior method, but each integrator has advantages in different regimes. For the quantum spin systems, the rank-adaptive and the fixed-rank BUG integrator performed well. Although the parallel BUG gives a rougher approximation of the solution, its potential lies in the context of high-performance computing. Its fully parallel structure combined with a large cluster allows for much more efficient simulations of quantum systems with many (say $d > 30$) particles. In the context of the Henon-Heiles potential, only the fixed-rank BUG appeared to produce stable and correct approximations. Since the rank-adaptive and the parallel BUG appeared to perform poorly, a further investigation of how the augmentation in those integrators influences the stability of the simulation would be of interest.

All presented quantum spin systems were one-dimensional systems. For many applications, like the presented quantum computers from the introduction, two- and three-dimensional lattices of particles are of interest. The tree tensor network format directly allows for simulations of such quantum systems and could be a subject of future research.

In section 6.2 we studied the influence of the tree structure on the tree ranks and the memory footprint and in chapter 3 on the operator representation. In both cases, we studied balanced binary trees and the tensor train format, which are a standard tool in computing quantum dynamics. However, there are many more possible tree structures, which could perform better in certain situations. To our knowledge, there is no rigorous mathematical analysis for this, nor clear heuristics for a large class of tree structures. Direct comparisons with the two-dimensional PEPS format, could lead to similar results as the comparison of the balanced binary tree and the tensor train format in one dimension.

All presented time integration schemes are first-order methods in time, cf. theorem 4.2, theorem 4.3 and theorem 5.2. The generalization of higher-order methods for matrices to tree tensor networks and generally the derivation of higher-order methods for tree tensor networks could be of interest to further research. Since the rank-adaptive BUG integrator often shows a second-order error behaviour when applied to non-stiff problems like the quantum spin systems, it would be interesting to see if one could prove a robust second-order error bound in this setting.

Bibliography

[Bac23] Markus Bachmayr. Low-rank tensor methods for partial differential equations. *Acta Numerica*, 32:1–121, 2023.

[BCFN18] Fabio Benatti, Federico Carollo, Roberto Floreanini, and Heide Narnhofer. Quantum spin chain dissipative mean-field dynamics. *Journal of Physics A: Mathematical and Theoretical*, 51(32):325001, Jun 2018.

[BH09] Dietrich Braess and Wolfgang Hackbusch. On the efficient computation of high-dimensional integrals and the approximation by exponential sums. In *Multiscale, Nonlinear and Adaptive Approximation*, pages 39–74. Springer, 2009.

[BJWM00] Michael H. Beck, Andreas Jäckle, Graham A. Worth, and Hans-Dieter Meyer. The multiconfiguration time-dependent hartree (mctdh) method: a highly efficient algorithm for propagating wavepackets. *Physics Reports*, 324(1):1–105, 2000.

[BK+15] Brett W. Bader, Tamara G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, February 2015.

[Bou23] Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

[BP02] Heinz-Peter Breuer and Francesco Petruccione. *The theory of open quantum systems*. Oxford University Press on Demand, 2002.

[BSU16] Markus Bachmayr, Reinhold Schneider, and André Uschmajew. Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. *Foundations of Computational Mathematics*, 16:1423–1472, 2016.

[CE22] Fabio Cassini and Lukas Einkemmer. Efficient 6d vlasov simulation using the dynamical low-rank framework ensign. *Computer Physics Communications*, 280:108489, 2022.

[CEKL24] Gianluca Ceruti, Lukas Einkemmer, Jonas Kusch, and Christian Lubich. A robust second-order low-rank bug integrator based on the midpoint rule. *BIT Numerical Mathematics*, 64(3):30, 2024.

[CGV23] Benjamin Carrel, Martin J Gander, and Bart Vandereycken. Low-rank parareal: a low-rank parallel-in-time integrator. *BIT Numerical Mathematics*, 63(1):13, 2023.

[CKL22] Gianluca Ceruti, Jonas Kusch, and Christian Lubich. A rank-adaptive robust integrator for dynamical low-rank approximation. *BIT Numer. Math.*, 2022.

[CKL24] Gianluca Ceruti, Jonas Kusch, and Christian Lubich. A parallel rank-adaptive integrator for dynamical low-rank approximation. *SIAM Journal on Scientific Computing*, 46(3):B205–B228, 2024.

[CKLS24] Gianluca Ceruti, Jonas Kusch, Christian Lubich, and Dominik Sulz. A parallel basis update and galerkin integrator for tree tensor networks. *arXiv preprint arXiv:2412.00858*, 2024.

[CKS24] Gianluca Ceruti, Daniel Kressner, and Dominik Sulz. Low-rank tree tensor network operators for long-range pairwise interactions. *arXiv preprint arXiv:2405.09952*, 2024.

[CL21] Gianluca Ceruti and Christian Lubich. An unconventional robust integrator for dynamical low-rank approximation. *BIT Numer. Math.*, 2021.

[CLS23] Gianluca Ceruti, Christian Lubich, and Dominik Sulz. Rank-adaptive time integration of tree tensor networks. *SIAM Journal on Numerical Analysis*, 61(1):194–222, 2023.

[CLW21] Gianluca Ceruti, Christian Lubich, and Hanna Walach. Time integration of tree tensor networks. *SIAM J. Numer. Anal.*, 59(1):289–313, 2021.

[CPGSV21] J Ignacio Cirac, David Perez-Garcia, Norbert Schuch, and Frank Verstraete. Matrix product states and projected entangled pair states: Concepts, symmetries, theorems. *Reviews of Modern Physics*, 93(4):045003, 2021.

[CV23] Benjamin Carrel and Bart Vandereycken. Projected exponential methods for stiff dynamical low-rank approximation problems. *arXiv preprint arXiv:2312.00172*, 2023.

[DDV00] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21:1253–1278, 2000.

[DEL19] Zhiyan Ding, Lukas Einkemmer, and Qin Li. Error analysis of an asymptotic preserving dynamical low-rank integrator for the multi-scale radiative transfer equation. *arXiv preprint arXiv:1907.04247*, 2019.

[Dir30] Paul AM Dirac. Note on exchange phenomena in the thomas atom. In *Mathematical proceedings of the Cambridge philosophical society*, volume 26, pages 376–385. Cambridge University Press, 1930.

[DK13] Sergey Dolgov and Boris Khoromskij. Two-level QTT-Tucker format for optimized tensor calculus. *SIAM J. Matrix Anal. Appl.*, 34(2):593–623, 2013.

[EG99] Yuli Eidelman and Israel Gohberg. On a new class of structured matrices. *Integral Equations Operator Theory*, 34(3):293–324, 1999.

[EJ21] Lukas Einkemmer and Ilon Joseph. A mass, momentum, and energy conservative dynamical low-rank scheme for the vlasov equation. *Journal of Computational Physics*, 443:110495, 2021.

[EL18] Lukas Einkemmer and Christian Lubich. A low-rank projector-splitting integrator for the vlasov–poisson equation. *SIAM Journal on Scientific Computing*, 40(5):B1330–B1360, 2018.

[EMP24a] Lukas Einkemmer, Julian Mangott, and Martina Prugger. A hierarchical dynamical low-rank algorithm for the stochastic description of large reaction networks. *arXiv preprint arXiv:2407.11792*, 2024.

[EMP24b] Lukas Einkemmer, Julian Mangott, and Martina Prugger. A low-rank complexity reduction algorithm for the high-dimensional kinetic chemical master equation. *Journal of Computational Physics*, 503:112827, 2024.

[FL18] Florian Feppon and Pierre FJ Lermusiaux. A geometric approach to dynamical model order reduction. *SIAM Journal on Matrix Analysis and Applications*, 39(1):510–538, 2018.

[Fre34] Jacov Frenkel. Wave mechanics: advanced general theory. 1934.

[GKT13] Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[Gra10] Lars Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM J. Matrix Anal. Appl.*, 31:2029–2054, 2010.

[Gro96] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[GVL13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations - 4th Edition*. Johns Hopkins University Press, Philadelphia, PA, 2013.

[Hac12] Wolfgang Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Springer, 2012.

[HLO+16] Jutho Haegeman, Christian Lubich, Ivan Oseledets, Bart Vandereycken, and Frank Verstraete. Unifying time evolution and optimization with matrix product states. *Phys. Rev. B*, 94(16):165116, 2016.

[HNS23a] Marlis Hochbruck, Markus Neher, and Stefan Schrammer. Dynamical low-rank integrators for second-order matrix differential equations. *BIT Numerical Mathematics*, 63:4, 2023.

[HNS23b] Marlis Hochbruck, Markus Neher, and Stefan Schrammer. Rank-adaptive dynamical low-rank integrators for first-order and second-order matrix differential equations. *BIT Numerical Mathematics*, 63(1):9, 2023.

[HNW93] Ernst Hairer, Syvert Norsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8. 01 1993.

[HRS12] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. On manifolds of tensors of fixed tt-rank. *Numerische Mathematik*, 120(4):701–731, 2012.

[HS23] Cory D Hauck and Stefan Schnake. A predictor-corrector strategy for adaptivity in dynamical low-rank approximations. *SIAM Journal on Matrix Analysis and Applications*, 44(3):971–1005, 2023.

[JBV+18] Jiasen Jin, Alberto Biella, Oscar Viyuela, Cristiano Ciuti, Rosario Fazio, and Davide Rossini. Phase diagram of the dissipative quantum ising model on a square lattice. *Phys. Rev. B*, 98:241108, Dec 2018.

[KB09] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51:455–500, 2009.

[KCEF22] Jonas Kusch, Gianluca Ceruti, Lukas Einkemmer, and Martin Frank. Dynamical low-rank approximation for burgers'equation with uncertainty. *International Journal for Uncertainty Quantification*, 12(5), 2022.

[KEC23] Jonas Kusch, Lukas Einkemmer, and Gianluca Ceruti. On the stability of robust dynamical low-rank approximations for hyperbolic problems. *SIAM Journal on Scientific Computing*, 45(1):A1–A24, 2023.

[KL07] Othmar Koch and Christian Lubich. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.*, 29(2):434–454, 2007.

[KL10] Othmar Koch and Christian Lubich. Dynamical tensor approximation. *SIAM J. Matrix Anal. Appl.*, 31:2360–2375, 2010.

[KLW16] Emil Kieri, Christian Lubich, and Hanna Walach. Discretized dynamical low-rank approximation in the presence of small singular values. *SIAM J. Numer. Anal.*, 54:1020–1038, 2016.

[KMR19] Daniel Kressner, Stefano Massei, and Leonardo Robol. Low-rank updates and a divide-and-conquer method for linear matrix equations. *SIAM J. Sci. Comput.*, 41(2):A848–A876, 2019.

[KNZ24] Yoshihito Kazashi, Fabio Nobile, and Fabio Zoccolan. Dynamical low-rank approximation for stochastic differential equations. *Mathematics of Computation*, 2024.

[KRS13] Vladimir Kazeev, Oleg Reichmann, and Christoph Schwab. Low-rank tensor structure of linear diffusion operators in the TT and QTT formats. *Linear Algebra Appl.*, 438(11):4204–4221, 2013.

[KS23] Jonas Kusch and Pia Stammer. A robust collision source method for rank adaptive dynamical low-rank approximation in radiation therapy. *ESAIM: Mathematical Modelling and Numerical Analysis*, 57(2):865–891, 2023.

[KT14] Daniel Kressner and Christine Tobler. Algorithm 941: `htucker`–a Matlab toolbox for tensors in hierarchical Tucker format. *ACM Trans. Math. Software*, 40(3):Art. 22, 22, 2014.

[Kus24] Jonas Kusch. Second-order robust parallel integrators for dynamical low-rank approximation. *arXiv preprint arXiv:2403.02834*, 2024.

[Lin76] Goran Lindblad. On the generators of quantum dynamical semigroups. *Commun. Math. Phys.*, 48(2):119–130, 1976.

[LO14] Christian Lubich and Ivan V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT Numer. Math.*, 54:171–188, 2014.

[LOV15] Christian Lubich, Ivan V Oseledets, and Bart Vandereycken. Time integration of tensor trains. *SIAM J. Numer. Anal.*, 53:917–941, 2015.

[LRSV13] Christian Lubich, Thorsten Rohwedder, Reinhold Schneider, and Bart Vandereycken. Dynamical approximation by hierarchical tucker and tensor-train tensors. *SIAM Journal on Matrix Analysis and Applications*, 34(2):470–494, 2013.

[LRY+24] Weitang Li, Jiajun Ren, Hengrui Yang, Haobin Wang, and Zhigang Shuai. Optimal tree tensor network operators for tensor network simulations: Applications to open quantum systems. *The Journal of Chemical Physics*, 161(5), 2024.

[LT21] Lin Lin and Yu Tong. Low-rank representation of tensor network operators with long-range pairwise interactions. *SIAM J. Sci. Comput.*, 43(1):A164–A192, 2021.

[Lub08] Christian Lubich. *From quantum to classical molecular dynamics: reduced models and numerical analysis.* Zurich Lectures in Advanced Mathematics. European Mathematical Society (EMS), Zürich, 2008.

[LVW18] Christian Lubich, Bart Vandereycken, and Hanna Walach. Time integration of rank-constrained Tucker tensors. *SIAM J. Numer. Anal.*, 56:1273–1290, 2018.

[MHM24a] Richard M Milbradt, Qunsheng Huang, and Christian B Mendl. Pytreenet: A python library for easy utilisation of tree tensor networks. *arXiv preprint arXiv:2407.13249*, 2024.

[MHM24b] Richard M Milbradt, Qunsheng Huang, and Christian B Mendl. State diagrams to determine tree tensor network operators. *SciPost Physics Core*, 7(2):036, 2024.

[MLC24] Robert Mattes, Igor Lesanovsky, and Federico Carollo. Long-range interacting systems are locally non-interacting. *arXiv preprint arXiv:2407.02141*, 2024.

[MRK20] Stefano Massei, Leonardo Robol, and Daniel Kressner. hm-toolbox: Matlab software for hodlr and hss matrices. *SIAM Journal on Scientific Computing*, 42(2):C43–C68, 2020.

[NAB24] Mohammad Hossein Naderi, Sara Akhavan, and Hessam Babaee. Cur for implicit time integration of random partial differential equations on low-rank matrix manifolds. *arXiv preprint arXiv:2408.16591*, 2024.

[NM02] Mathias Nest and Hans-Dieter Meyer. Benchmark calculations on high-dimensional henon–heiles potentials with the multi-configuration time dependent hartree (mctdh) method. *The Journal of Chemical Physics*, 117(23):10499–10505, 2002.

[NT24] Fabio Nobile and Thomas Trigo Trindade. Petrov-galerkin dynamical low rank approximation: Supg stabilisation of advection-dominated problems. *arXiv preprint arXiv:2407.01404*, 2024.

[O'b07] Jeremy L O'brien. Optical quantum computing. *Science*, 318(5856):1567–1570, 2007.

[Ose11] Ivan V Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, 2011.

[PGVWC07] David Perez-García, Frank Verstraete, Michael M Wolf, and J Ignacio Cirac. Matrix product state representations. *Quantum Information and Computation*, 7(5-6):401–430, 2007.

[PKS+19] Sebastian Paeckel, Thomas Köhler, Andreas Swoboda, Salvatore R. Manmana, Ulrich Schollwöck, and Claudius Hubig. Time-evolution methods for matrix-product states. *Annals of Physics*, 411:167998, 2019.

[PMCV10] Bogdan Pirvu, Valentin Murg, J Ignacio Cirac, and Frank Verstraete. Matrix product operator representations. *New Journal of Physics*, 12(2):025012, 2010.

[PMF20] Zhuogang Peng, Ryan G McClarren, and Martin Frank. A low-rank method for two-dimensional time-dependent radiation transport calculations. *Journal of Computational Physics*, 421:109735, 2020.

[RLJS20] Jiajun Ren, Weitang Li, Tong Jiang, and Zhigang Shuai. A general automatic method for optimal construction of matrix product operators using bipartite graph theory. *The Journal of Chemical Physics*, 153(8), 2020.

[Sch11] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011.

[SCK24] Axel Séguin, Gianluca Ceruti, and Daniel Kressner. From low-rank retractions to dynamical low-rank approximation and back. *BIT Numerical Mathematics*, 64(3):25, 2024.

[SHNT23] Jonathan Schmidt, Philipp Hennig, Jörg Nick, and Filip Tronarp. The rank-reduced kalman filter: Approximate dynamical-low-rank filtering in

high dimensions. *Advances in Neural Information Processing Systems*, 36:61364–61376, 2023.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[SLC+24] Dominik Sulz, Christian Lubich, Gianluca Ceruti, Igor Lesanovsky, and Federico Carollo. Numerical simulation of long-range open quantum many-body dynamics with tree tensor networks. *Physical Review A*, 109(2):022420, 2024.

[SMC+23] Philipp Seitz, Ismael Medina, Esther Cruz, Qunsheng Huang, and Christian B Mendl. Simulating quantum circuits using tree tensor networks. *Quantum*, 7:964, 2023.

[Sti73] R B Stinchcombe. Ising model in a transverse field. I. Basic theory. *J. Phys. C: Solid State Physics*, 6(15):2459–2483, 1973.

[Sul24] Dominik Sulz. *Numerical test cases for Time Integration in Quantum Dynamcis using Tree Tensor Networks*, 2024. `https://github.com/DominikSulz/PhD_thesis_simulations`.

[SWM10] Mark Saffman, Thad G. Walker, and Klaus Mølmer. Quantum information with Rydberg atoms. *Rev. Mod. Phys.*, 82:2313–2363, Aug 2010.

[SZCT24] Dayana Savostianova, Emanuele Zangrando, Gianluca Ceruti, and Francesco Tudisco. Robust low-rank training via approximate orthonormal constraints. *Advances in Neural Information Processing Systems*, 36, 2024.

[SZK+22] Steffen Schotthöfer, Emanuele Zangrando, Jonas Kusch, Gianluca Ceruti, and Francesco Tudisco. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. *Advances in Neural Information Processing Systems*, 35:20051–20063, 2022.

[Tob12] Christine Tobler. Low-rank tensor methods for linear systems and eigenvalue problems, 2012. PhD thesis no. 20320 ETH Zürich.

[Tuc66] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[UV13] André Uschmajew and Bart Vandereycken. The geometry of algorithms using hierarchical tensors. *Linear Algebra and its Applications*, 439(1):133–166, 2013.

[UZ24] André Uschmajew and Andreas Zeiser. Dynamical low-rank approximation of the vlasov–poisson equation with piecewise linear spatial boundary. *BIT Numerical Mathematics*, 64(2):19, 2024.

[VDS+16] Nico Vervliet, Otto Debals, Laurent Sorber, Marc Van Barel, and Lieven De Lathauwer. Tensorlab 3.0, Mar. 2016. Available online.

[VMC08] Frank Verstraete, Valentin Murg, and J Ignacio Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in physics*, 57(2):143–224, 2008.

[WT03] Haobin Wang and Michael Thoss. Multilayer formulation of the multiconfiguration time-dependent Hartree theory. *J. Chem. Phys.*, 119(3):1289–1299, 2003.

[XCGL10] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, and Xiaoye S. Li. Fast algorithms for hierarchically semiseparable matrices. *Numer. Linear Algebra Appl.*, 17(6):953–976, 2010.

[XXCB14] Yuanzhe Xi, Jianlin Xia, Stephen Cauley, and Venkataramanan Balakrishnan. Superfast and stable structured solvers for Toeplitz least squares via randomized sampling. *SIAM J. Matrix Anal. Appl.*, 35(1):44–72, 2014.

[ZSC+23] Emanuele Zangrando, Steffen Schotthöfer, Gianluca Ceruti, Jonas Kusch, and Francesco Tudisco. Rank-adaptive spectral pruning of convolutional layers during training. *arXiv preprint arXiv:2305.19059*, 2023.