Technische Universität München
TUM School of Natural Sciences

Advancing Tensor Network Simulations

—

Hardware Acceleration, Stable Optimization, and Symmetries

Jakob Unfried

Vollständiger Abdruck der von der TUM School of Natural Sciences der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Jonathan J. Finley, Ph.D.

Prüfende der Dissertation:

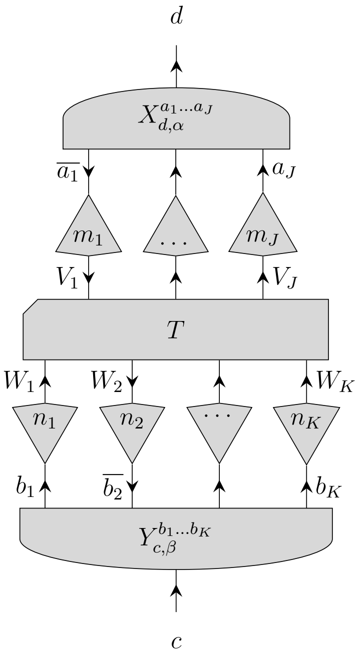
1. Prof. Dr. Frank Pollmann
2. Prof. Dr. Johannes Knolle

Die Dissertation wurde am 27.09.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Natural Sciences am 09.10.2024 angenommen.

Advancing Tensor Network Simulations
—
Hardware Acceleration, Stable Optimization, and Symmetries

Dissertation

Jakob Unfried



Chair of Theoretical Solid-State Physics
TUM School of Natural Sciences, Physics Department
Technical University of Munich

Abstract

Strongly correlated quantum many-body systems can exhibit a wide variety of phenomena, such as high-temperature superconductivity or the fractional quantum Hall effect. Numerical simulations of these systems are challenging due to the exponential scaling of the dimension of the many-body Hilbert space with system size. Tensor network methods offer powerful algorithms to approach these systems numerically, extracting predictions and measurable signatures from candidate theories for comparison with experiments. We discuss several algorithmic improvements for tensor network algorithms. In particular, we propose and benchmark (i) a modified truncation step in matrix product state simulations that enables the full benefit of hardware acceleration by avoiding singular value decompositions. We propose and benchmark (ii) a gradient-based approach for the optimization of tensor networks for finite two-dimensional systems, enabling ground state search and time evolution. Finally, we introduce (iii) a framework to incorporate nonabelian symmetries, as well as fermionic or anyonic exchange statistics into tensor network simulations on the tensor level.

Zusammenfassung

Stark korrelierte Quanten-Vielteilchensysteme können eine Vielzahl von interessanten Phänomenen aufweisen, wie z.B. Hochtemperatursupraleitung oder den fraktionalen Quanten-Hall-Effekt. Die numerische Simulation solcher Systeme stellt jedoch eine Herausforderung dar, da die Dimension des Vielteilchen-Hilbertraums mit der Systemgröße exponentiell anwächst. Tensornetzwerk-Methoden bieten leistungsstarke Algorithmen, um diese Systeme numerisch zu untersuchen, und aus Kanditaten einer Theorie des Systems Vorhersagen von Messgrößen zu extrahieren, und Vergleich mit Experimenten zu ermöglichen. In dieser Arbeit stellen wir mehrere algorithmische Verbesserungen für Tensornetzwerk-Algorithmen vor. Insbesondere (i) schlagen wir einen modifizierten Trunkierungsschritt in Matrixproduktzustand-Simulationen vor, der durch die Vermeidung von Singulärwertzerlegungen die volle Hardwarebeschleunigung, z.B. von Grafikkarten ermöglicht, (ii) entwickeln und testen wir einen gradientenbasierten Ansatz zur Optimierung von Tensornetzwerken für endliche zweidimensionale Systeme, der sowohl die Simulation von Grundzuständen, als auch von Zeitentwicklung erlaubt, und (iii) stellen wir ein mathematischen Rahmen vor, um nichtabelsche Symmetrien sowie fermionische oder anyonische Austauschstatistiken auf der Tensorebene in Tensornetzwerk-Simulationen zu integrieren.

Contents

List of Publications

Parts of this thesis also appear in the following works:

[1] J. Unfried, J. Hauschild, and F. Pollmann, *Fast time evolution of matrix product states using the QR decomposition*, *Physical Review B*, vol. 107, no. 15, p. 155 133, Apr. 2023. DOI: `10.1103/PhysRevB.107.155133`.

[2] J. Unfried, L. Vanderstraeten, M. Knap, and F. Pollmann, *Gradient-based optimization for dynamics of Projected Entangled Pair States*, *In preparation*, 2024.

[3] J. Hauschild, J. Unfried, *et al.*, *Tensor Network Python (TeNPy) version 1*. DOI: `10.48550/arXiv.2408.02010`.

and many the discussed concepts are implemented in the software library:

[4] J. Hauschild, J. Unfried, F. Pollmann, and T. developers, *Tensor Network Python (TeNPy)*DOI: : https://github.com/tenpy/tenpy.

The QR-based truncation scheme and resulting QR-based TEBD algorithm presented in chapter 3 have appeared in Ref. [1].

The global gradient-based optimization scheme for finite PEPS, introduced in chapter 4 is part of a publication [2] that is, at the time of writing, in the final stages of preparation. The introduction to PEPS in section 2.4 is also largely based on that manuscript.

The TeNPy software package [4] for tensor network simulations in python is co-maintained by the author, and contains an implementation of the QR-based TEBD algorithm of chapter 3. It has recently seen a version 1 release, accompanied by the article [3]. The framework for nonabelian symmetries, as well as fermionic or anyonic degrees of freedom developed in chapter 5 is the basis for current active development of a next version of the package.

Chapter 1

Introduction

Understanding the rich phenomenology of strongly correlated quantum many-body systems, such as e.g. high-temperature superconductivity [5, 6] or the fractional quantum Hall effect [7–9] remains an open challenge in condensed matter physics. One key step in this quest is extracting predictions of measurable quantities from candidate theories, allowing for a connection to experiments. Due to the strong correlations, even in idealized models, such as the Hubbard model [10] and the related t-J model, which are proposed as minimal models to understand superconductivity [11–13], this is not possible analytically without further simplification or approximation, and thus requires numerical methods.

The main challenge in numerical approaches to analyzing these quantum many-body models is the *curse of dimensionality*, meaning that the dimension of the many-body Hilbert space grows exponentially with system size. Thus, direct approaches – commonly dubbed exact diagonalization (ED) – which find the ground state of a Hamiltonian by directly tackling the eigenvalue problem numerically, are limited to small system sizes. Quantum Monte Carlo methods [14] do not operate on these exponentially dimensional objects and instead stochastically sample the quantities of interest such that large systems can be simulated efficiently. Thus, if the sign problem [15] can be addressed, Monte Carlo methods offer arguably the most stable and efficient numerical approach to simulating quantum many-body systems. They encounter difficulties, however, in fermionic or geometrically frustrated settings and simulation of real-time dynamics. In these settings, variational methods and, in particular, tensor network methods are usually the most stable numerical approach to simulating these systems. In this thesis, we focus on tensor network methods.

Conventional wisdom in the community is that the key property to identify the correct class of tensor network to approximate a given target state is its entanglement. This correspondence between entanglement structure and variational power is exact for matrix product states (MPSs), where it is proven that ground states of gapped, local 1D Hamiltonians fulfill the area law of entanglement [16] and that any area law state can be efficiently approximated as an MPS [17, 18]. For the other classes of tensor network ansaetze and corresponding models, analogous statements are not as straightforward to establish rigorously but are believed to hold for most relevant models. This includes tree tensor networks (TTNs) [19–21] or the multiscale entanglement renormalization ansatz (MERA) [22–24] in one or more dimensions, which can capture the scale invariance and the entanglement structure of critical states, with a logarithmic correction and have indeed been found to describe some critical systems well. A generalization of MPS to higher dimensions gives the projected entangled pair state (PEPS) [25, 26]. While the correspondence between ground states of local gapped Hamiltonians and states that can be efficiently approximated by tensor product state (TPS) is believed to generalize to higher dimensions, at least for a wide class of physically relevant systems, establishing it both rigorously and on general terms was not possible so far.

While the entanglement structure of the target state may establish that it can *in principle* be well-approximated by a tensor network state (TNS) of the chosen structure, this is only half of the story, as we also need an algorithm to efficiently find this good approximation within the variational manifold. For MPS in 1D, this has been realized by e.g. the density matrix renormalization group (DMRG) [27, 28] and variational uniform matrix product state (VUMPS) [29] ground state search algorithms, which exploit the canonical form [30, 31] of MPS – a particularly convenient choice to fix the internal gauge degrees of freedom. The expectation of finding similar success using natively higher-dimensional TNS such as PEPS, however, has not been met, and PEPS simulations are typically less stable and more challenging conceptually and computationally.

In this thesis, we pursue three avenues to push the performance of tensor network simulations, extending the limits of what is accessible to them. Exploiting (i) hardware acceleration, such as e.g. the power of graphics processing units (GPUs) or dedicated tensor processing units (TPUs) is a promising avenue for pushing the boundaries of what is accessible to TNS simulations [32–34]. As such, tensor networks follow in the footsteps of neural networks, where hardware acceleration played a central role in the rise of artificial intelligence and machine learning tools. From an algorithmic point of view, this requires using linear algebra routines that are efficient on GPU, and in particular, to avoid the standard singular value decomposition (SVD), which is inefficient on GPUs. Next (ii), we explore global gradient-based optimization methods that recently gained traction as a more robust way to optimize PEPS, and in particular for ground state search on infinite systems, in new algorithmic settings, such as dynamics of finite PEPS. Lastly (iii), exploiting symmetries of the model, which is a well-established technique to improve the accuracy and performance of tensor network algorithms [35, 36]. Symmetric states can be targeted by constructing them as a variational tensor network of symmetric tensors. These symmetric tensors require fewer free parameters than general tensors, such that storing them requires less memory and operating on them requires fewer central processing unit (CPU) operations, increasing performance. Additionally, enforcing the conservation of a symmetry can increase the accuracy of the simulation, allows targeting specific charge sectors explicitly, and gives access to symmetry-resolved data.

The thesis is structured as follows.

In chapter 2, we review tensor network methods, introducing relevant concepts, notation, and the established TNS algorithms. In particular, we discuss the connection between entanglement and the variational power of common classes of TNS. We introduce the class of MPS, their isometric and canonical form, and the DMRG, time evolving block decimation (TEBD) and matrix operator based time evolution (MPO evolution) algorithms. We discuss tensor networks in higher-dimensional systems, mainly focusing on PEPS for 2D systems. We introduce the ansatz, approximate contraction methods, and briefly summarize common algorithms. We discuss how to exploit symmetries in TNS simulations in terms of the block-sparse structure that a symmetry imposes on tensors, focusing on abelian symmetry groups for concreteness. We briefly introduce the Tensor Network Python (TeNPy) python package that offers MPS simulations, exploiting abelian symmetry groups.

In chapter 3, we develop techniques to accelerate truncation steps in TNS algorithms by replacing the truncated SVD that is commonly used to renormalize the bond dimension of a tensor network with other low-rank factorization routines. We first focus on one particular routine developed in a previous publication [1] in the context of the TEBD algorithm for MPS time evolution, which we dubbed the QR decomposition (QR)-based truncation. We then discuss how this approach is related to randomized linear algebra and, in particular, can be understood as a modified version of a randomized singular value decomposition (rSVD) that is particularly suited for the MPS context. We propose a best-of-both-worlds synthesis of the truncation algorithm before showing benchmark results of the QR-based truncation routine.

In chapter 4, we propose an approach to the global, gradient-based optimization of finite PEPS, motivated by the success of similar methods for infinite systems [37–39], using automatic differentiation. We study the interplay of numerical gradient evaluation with the approximation methods needed to evaluate the cost function and the pathologies that we find arise. These pathological optimization trajectories optimize the approximately evaluated cost function not by optimizing the exact expression for the cost but rather by causing the approximation to become uncontrolled. We propose to remedy this by using the approximate contraction methods to evaluate an exact expression for the gradient instead of following the scheme of automatic differentiation, which computes the derivative of the approximation. We formulate a ground state search algorithm, as well as a time stepper, and showcase benchmark results, simulating the 2D quantum transverse field Ising model, computing ground states, and extracting the dynamical spin structure factor from quench dynamics.

In chapter 5, we introduce strategies to enforce nonabelian symmetries in tensor networks on the tensor level, based on fusion trees. We give a detailed pedagogical introduction to the underlying mathematical framework – the theory of monoidal categories, providing side-by-side an intuitive and a more rigorous perspective, with a common graphical language. This categorical approach to symmetries allows the machinery to be applied to tensors that intrinsically have the statistics of fermionic or anyonic excitations and can thus be used to build tensor network representations or approximations of fermionic/anyonic many-body states. We identify the free parameters of a symmetric tensor and derive in detail how to perform common linear algebra routines on them, such as combining, splitting, or re-arranging legs, pairwise contraction, and factorizations. This is the basis for a new version of TeNPy, currently under active development, with a working prototype developed by the author.

We conclude with a summary, a discussion of common themes throughout the chapters, and an outlook regarding future directions in chapter 6.

In appendix A, we provide the topological data of common symmetries. This is the data required to use the given symmetry in the framework of chapter 5, to represent and operate on tensors that have this symmetry. This includes a review a representation theory in section A.1, and data for the group symmetries $\mathbb{Z}_N$, U(1) and SU(2) in sections A.2-A.4 respectively, for fermionic grading in section A.5, for Fibonacci anyons in section A.6, and how to combine multiple symmetries in section A.7.

In appendix B, we provide derivations for the autodiff formulae stated in section 4.2.

The author emphasizes the benefits of open-source culture in science and provides all code associated with this thesis publicly on GitHub[1].

[1] https://github.com/Jakob-Unfried/phd_thesis

Chapter 2

Review of tensor network methods

Tensor networks have a long history, and related concepts were independently discovered from multiple perspectives, traced in detail, e.g. in [40, Sec. I.B]. The first works that can be considered precursors to the modern tensor network perspective dealt with classical statistical mechanics problems [41, 42]. First applications of related ideas in the realm of quantum mechanics, as a wavefunction ansatz [43, 44] led to the development of the class of finitely correlated states (FCSs) [45–47], which are closely related to MPSs. The reformulation of the density matrix renormalization group (DMRG) [48, 49] – an algorithm designed to study ground states of 1D systems – as a variational ground state search in the class of MPSs [27, 28, 46, 50] established TNS methods in their modern form. Various technical improvements of the algorithm have since pushed the boundaries of what can be efficiently simulated using MPSs. For example, exploiting abelian [35, 51] or nonabelian [36, 52, 53] symmetries, representation in hybrid real and momentum spaces [54, 55], and density matrix perturbations [56, 57] have improved convergence, accuracy and performance of DMRG simulations. A formulation for translationally invariant infinite matrix product state (iMPS) allows to study the thermodynamic limit directly, using infinite DMRG [58] or VUMPS [29] methods, as well as excitations on top of the ground states using tangent space methods [30, 31, 59]. Beyond ground states, thermal states can be simulated using purification methods [60, 61], and real-time evolution [62] allows access to e.g. transport properties and non-equilibrium phenomena. In the field of quantum computing, MPS methods can serve as a benchmark [63, 64] or a simulation of the quantum device itself [65, 66].

Beyond MPS, various other tensor network ansaetze with different connectivity of the tensors have been developed to address different settings and fall under the umbrella term of tensor network state (TNS). This includes TTNs [20] and the MERA [22, 24] for studying critical systems, originally in 1D, but readily generalized to higher dimensions [67, 68], as well as PEPS [25, 26], which are the direct generalization of MPS to higher-dimensional systems. Various algorithms for optimizing variational PEPS ansaetze in both finite and infinite systems range from full update (FU) [69] to gradient-based methods [37–39], and many more. These natively higher-dimensional approaches, however, lack many of the favorable properties of

MPS in 1D, such as prominently the canonical form. While attempts have been made at establishing canonical forms [70, 71] and fixing the gauge freedom [72] of PEPS, they fall short of the advantageous properties of the canonical form in MPS. This discrepancy goes so far that MPS simulations are often the preferred way of studying 2D systems by mapping a thin stripe or thin cylinder to a chain at the cost of increasing the range of couplings in the Hamiltonian.

In this chapter, we provide a pedagogical introduction to many aspects of tensor network simulations. This compilation covers only select topics, and we refer the interested reader to the review articles [28, 40, 62, 73, 74]. In section 2.1, we introduce basic concepts of (bipartite) entanglement, the area law, and how the entanglement structure of the target state determines the appropriate TNS ansatz. In section 2.2, we briefly introduce the graphical notation of tensor networks and basic operations on tensors before discussing MPSs in section 2.3. We cover the canonical form, emphasizing its weaker, more general version, the isometric form, and discuss the DMRG ground state search, as well as TEBD and MPO evolution methods for simulating dynamics. We discuss TNS approaches to higher-dimensional (more than one spatial dimension) systems in section 2.4, with a focus on PEPS methods. In section 2.5, we review in detail the mathematical framework for exploiting (abelian) symmetries in tensor networks as a basis for discussing decomposition routines for symmetric tensors in chapter 3, and as the base case for discussion of more general symmetries in chapter 5. Finally, we highlight the TeNPy software package for tensor network simulations in section 2.6.

2.1 Entanglement

Entanglement is a key measure to quantify the complexity of strongly correlated quantum states. It can be understood as a resource in quantum state preparation [75–77], allows a classification of critical states, e.g. in terms of the central charge in the presence of conformal invariance [78–80] and exhibits characteristic signatures of topological order [81–83]. The entanglement structure of a given target state is the basis for approximating it with a tensor network ansatz.

2.1.1 Schmidt decomposition and Bipartite Entanglement

The most common and most accessible measure for entanglement is given by the bipartite entanglement entropy. Consider a quantum system described by a Hilbert space $\mathcal{H}$ and a fixed bipartition $\mathcal{H} = \mathcal{H}_L \otimes \mathcal{H}_R$, which is usually chosen as a *spatial* bipartition, i.e. $\mathcal{H}_L$ describes the degrees of freedom within a certain spatial region of the whole system and $\mathcal{H}_R$ its complement. The bipartite entanglement entropy

$$S_{L,R} = S_{\mathrm{vN}}(\rho_L) \tag{2.1}$$

of a pure state $|\psi\rangle \in \mathcal{H}$ is then given by the von-Neumann entropy $S_{\mathrm{vN}}(\rho_L) = -\mathrm{Tr}\left(\rho_L \log \rho_L\right)$ of the reduced density matrix

$$\rho_L = \mathrm{Tr}_R\left(|\psi\rangle\langle\psi|\right). \tag{2.2}$$

Note that the definition is symmetric, i.e. $S_{L,R} = S_{\mathrm{vN}}(\rho_B)$, which is particularly apparent using the Schmidt decomposition, which allows direct access to the entanglement entropy.

A Schmidt decomposition with respect to the bipartition $\mathcal{H} = \mathcal{H}_L \otimes \mathcal{H}_R$ consists of two orthonormal sets $\{|L_\alpha\rangle\} \subset \mathcal{H}_L$, and $\{|R_\alpha\rangle\} \subset \mathcal{H}_R$ of states on either subsystem, known as the left and right *Schmidt states*, together with non-negative real numbers Λ_α, the *Schmidt values*, such that

$$|\psi\rangle = \sum_{\alpha=1}^{k} \Lambda_\alpha \, |L_\alpha\rangle \otimes |R_\alpha\rangle \,, \tag{2.3}$$

where $k = \min(\dim \mathcal{H}_L, \dim \mathcal{H}_R)$. It is guaranteed to exist for any state $|\psi\rangle$ and is unique up to the relative phase of the Schmidt states, $(L_\alpha, R_\alpha) \mapsto (\mathrm{e}^{\mathrm{i}\phi_\alpha} L_\alpha, \mathrm{e}^{-\mathrm{i}\phi_\alpha} R_\alpha)$, assuming non-degenerate Schmidt values. The Schmidt decomposition allows direct access to the reduced density matrices and the bipartite entanglement entropy, as

$$\rho_L = \sum_{\alpha=1}^{k} \Lambda_\alpha^2 \, |L_\alpha\rangle\langle L_\alpha| \quad \text{and} \quad \rho_R = \mathrm{Tr}_L \left(|\psi\rangle\langle\psi| \right) = \sum_{\alpha=1}^{k} \Lambda_\alpha^2 \, |R_\alpha\rangle\langle R_\alpha| \tag{2.4}$$

have the same set $\{\Lambda_\alpha^2 | \Lambda_\alpha \neq 0\}$ of nonzero eigenvalues, such that

$$S_{L,R} = -\sum_\alpha \Lambda_\alpha^2 \log \Lambda_\alpha^2. \tag{2.5}$$

2.1.2 Area law

Let us now assume a many-body system consisting of N sites, each with a Hilbert space H, such that the many-body Hilbert space is $\mathcal{H} = \bigotimes_{n=1}^{N} H$. Further, consider a spatial bipartition into L sites to the left and $R = N - L$ sites to the right such that $\mathcal{H}_L = \bigotimes_{n=1}^{L} H$ and $\mathcal{H}_R = \bigotimes_{n=L+1}^{N} H$, and assume that $L \leq R$. Now, for a generic state, the Schmidt spectrum w.r.t. such a bipartition is roughly constant, that is $\Lambda_\alpha \approx 1/\sqrt{k} = \text{const.}$ such that $S \approx \log k \approx L$. It has been shown [84] that for $L = R = N/2$ we find on averaging over all states of the system an entropy of $S = L \log d - \frac{1}{2} \sim L$. This scaling $S \sim L$ with the size/volume L of the subsystem for $L \gg 1$ is commonly called *volume law*.

This is in contrast to *area law* states, for which entanglement scales only with the volume ("area" as a $(D-1)$-dimensional measure) of the boundary between the two subsystems. In 1D, this is a constant, and it can be shown that these area law states correspond exactly to the ground states of gapped, local Hamiltonians. The intuition here is that entanglement or correlations are a finite resource, and minimizing the energy of the local terms in the Hamiltonian prefers building correlations between spatially close degrees of freedom over correlations between distant sites. As a result, the ground state has a correlation length ξ, and only those sites in a finite strip of width $\sim \xi$ to either side of the boundary contribute to the entanglement between the subsystems. Thus, for large enough subsystems $L \gg \xi$, we find that the entropy

$S \sim |\partial L|$ scales with the volume of the boundary ∂L of the subsystem. While this heuristic carries over to higher dimensions and may be realized in this way for certain models, the correspondence between area law states and ground states of local Hamiltonians is not exact.

For area law states, the distribution of Schmidt values in the decomposition (2.3) has its weight concentrated in a small number of Schmidt values. As a result, we can efficiently approximate area law states by truncating the Schmidt spectrum. In particular, the number χ of Schmidt values that are required to achieve a target error ϵ, i.e. such that

$$\left\| |\psi\rangle - \sum_{\alpha=1}^{\chi} \Lambda_\alpha |L_\alpha\rangle \otimes |R_\alpha\rangle \right\| \leq \epsilon \tag{2.6}$$

is finite and *independent of (sub-)system size.*

For a given tensor network geometry, an upper bound for the entanglement scaling can easily be derived. Find a minimal[1] cut through the tensor network that splits it into the bipartition of interest. This cut gives a factorization of the many-body wave function with rank $K = \prod_i \chi_i$ given by the total dimension of the cut virtual legs, each with bond dimension χ_i. Thus, we have at most K non-zero Schmidt values for the bipartition, and thus at most $S = \log K = \sum_i \log \chi_i$. For MPS, we only need to cut a single bond and find an area law of $S \leq \log \chi = \text{const.}$. For MERA in 1D, we need to cut a number of bonds that grows logarithmically with subsystem size, giving us logarithmic corrections $S \leq \log L \log \chi$, assuming all virtual bonds have the same dimension χ. For TPS in higher dimensions, we find an area law $S \leq |\partial L| \log \chi$, assuming all virtual bonds have the same dimension χ.

For MPS in 1D, any area law state can be efficiently approximated by an MPS with a bounded bond dimension, *independent of system size.* This is because the canonical form of MPS connects a truncation of MPS bond dimension on a single bond to a truncation of a Schmidt decomposition (2.6). It can be shown that repeating this truncation for the remaining bonds results in an error that remains controllable, such that approximation within a fixed error threshold is still possible at finite bond dimension independent of (sub-)system size.

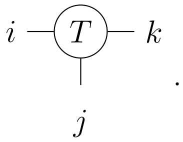
2.2 Tensors and diagrammatic notation

The building blocks of TNSs are tensors. From a mathematical perspective, tensors are elements of a tensor product space, e.g. $T \in \mathbb{C}^5 \otimes \mathbb{C}^3 \otimes \mathbb{C}^2$ is a three-leg tensor. More straight-forwardly we can think of tensors as multi-dimensional arrays of numbers, generalizing the notion of a matrix A with entries A_{ij} to more (or fewer) than two indices, giving us tensors such as e.g. T with entries T_{ijk}. There is a graphical notation for tensors and tensor networks that is established in the community.

[1] Any cut gives an upper bound. Choose a minimal cut for the tightest possible upper bound.

Tensors are represented with shapes with legs, which represent the indices, e.g.

$$T_{ijk} \quad =:$$

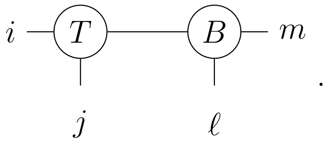


(2.7)

This involves a fixed convention regarding which leg points in which direction.

Tensor contraction, that is a sum over a shared index, is represented by connecting the legs, that is, e.g.

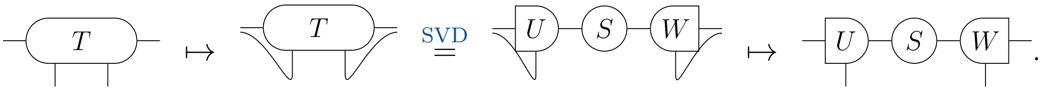
$$\sum_k T_{ijk} B_{k\ell m} \quad =:$$



(2.8)

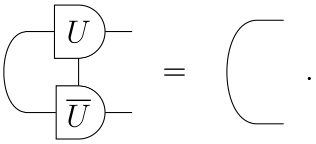
The labels i, j, ℓ, m are typically omitted in equations where both sides are given diagrammatically, such as e.g. in (2.9), and indices are identified not by matching index, but by matching position and orientation of the leg.

Another ubiquitous step in tensor networks is to apply matrix decompositions to tensors. This is done by first reshaping to a matrix, meaning combining indices into two groups $T_{ijk\ell} = T_{(ij),(k\ell)}$ and understanding the result as a matrix, and performing standard factorization, such as e.g. an SVD on that matrix. Finally, we ungroup the legs of the factors, e.g. as $U_{(ij),m} = U_{ijm}$ to obtain a tensor factorization, which shares the properties of the SVD



(2.9)

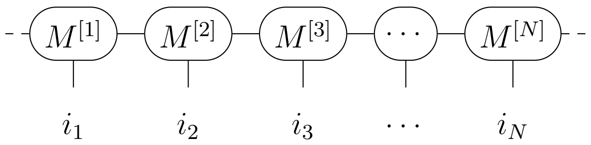
We use boxes with one rounded side for the isometric tensors U, W, where a left (right) isometry, such as, e.g., U (W), is an isometric map from the group of the left (right) and a bottom leg to the right (left) leg. In particular, this means



(2.10)

2.3 Matrix Product States (MPS) in one dimension

A (finite) matrix product state (MPS) is a quantum state on a chain of N sites, where the coefficients of the wavefunction in computational bases $\{|i_n\rangle\}$ on each site n, is given by the following tensor network

$|\psi\rangle \;:=\; \sum_{i_1,\ldots,i_N}$  $|i_1,\ldots,i_N\rangle$, (2.11)

where the dashed lines indicate a trivial index that can only take on a single value.

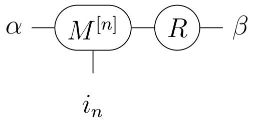
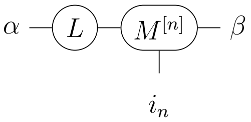
For fixed physical indices $i_1, \ldots, i_N$, the three-leg tensors $M^{[2]}$ reduce to a matrix $M^{[2]i_2}$ of the two remaining *virtual* indices, such that the wavefunction coefficients

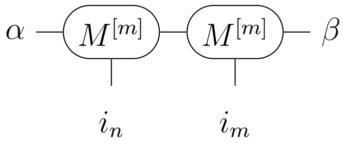
$$\langle i_1, \ldots, i_N|\psi\rangle = \sum_{\alpha_1,\ldots,\alpha_{N-1}} (M^{[1]i_1})_{1,\alpha_1}(M^{[2]i_2})_{\alpha_1,\alpha_2}\ldots(M^{[N]i_N})_{\alpha_{N-1},1} \tag{2.12}$$

are given as a product of matrices, which coins the name of the ansatz. The *bond dimension* χ_n of the n-th bond between sites $n, n+1$ is given by the dimension of the contracted index α_n between the respective tensors $M^{[n]}, M^{[n+1]}$. The *maximal bond dimension* $\chi = \max_n \chi_n$ is also referred to simply as "the bond dimension of the MPS".

In this section, we discuss several algorithmic aspects as basics for the later chapters of this thesis. We cover the gauge freedom of MPS and how to exploit it to enforce special properties, namely the isometric form as a weaker requirement or the canonical form as a stronger special case, the DMRG algorithm for ground state search, as well as the TEBD and MPO evolution algorithms for simulating dynamics. We do not cover the time dependent variational principle (TDVP) algorithm [85, 86] for time evolution, infinite MPS [87], the VUMPS algorithm for ground state search [29], or tangent space methods for excitations [30, 31, 59], and refer the interested reader to the given references or reviews [40, 73].

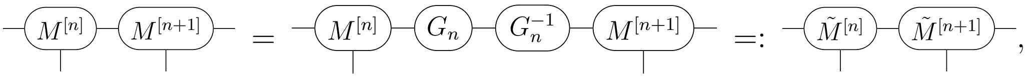
Throughout the section, we employ the following slight abuse of notation for brevity. We use dot products to indicate the contraction of virtual MPS legs and imply the behavior of the physical legs, which remain untouched and in order of occurrence. Any symbol with a superscript in square brackets is an MPS tensor and is understood to have three indices α, i, β with i being the physical index, while other symbols are assumed to have only two virtual indices α, β. If neither symbol has a physical leg, the dot product denotes the matrix product $(R \cdot L)_{\alpha,\beta} = \sum_\gamma R_{\alpha\gamma}L_{\gamma\beta}$. If there is one or two physical indices, we think of them as follows.

$(M^{[n]} \cdot R)_{\alpha i_n \beta} \;:=$  ; $(L \cdot M^{[n]})_{\alpha i_n \beta} \;:=$ 

$(M^{[n]} \cdot M^{[m]})_{\alpha i_n i_m \beta} \;:=$  (2.13)

2.3.1 Gauge fixing, isometric form and canonical form

A common feature of TNS is a gauge freedom on every virtual bond, meaning there are many choices for the tensors in a tensor network, such that the physical quantum state is the same. We may insert a resolution of identity $\mathbb{1} = G_n G_n^{-1}$ in terms of any invertible matrix $G_n \in \mathrm{GL}(\chi_n)$ on the n-th virtual bond of dimension χ_n, and absorb it into the respective tensors, that is

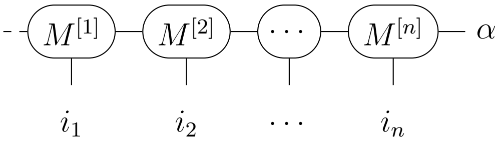


(2.14)

such that clearly the quantum state (2.11) remains unchanged.

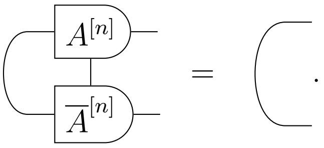
We can use this gauge freedom to enforce desirable properties on the tensors of the MPS. In particular, consider the following family of states on the subsystem left of bond n, indexed by $\alpha = 1, \dots, \chi_n$

$$\left|L_\alpha^{(n)}\right\rangle = \sum_{i_1,\dots,i_n}$$



$$|i_1, \dots, i_n\rangle \ . \qquad (2.15)$$

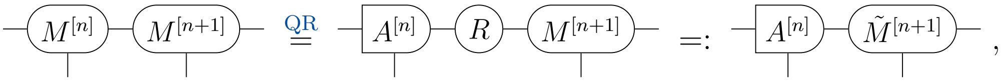
Let us assume that those states span a χ_n-dimensional subspace, since otherwise, the MPS could have been written down with a smaller bond dimension, namely the dimension of the span. A gauge transformation (2.14) acts on them as $|L_\alpha^{(n)}\rangle \mapsto |\tilde{L}_\alpha^{(n)}\rangle = \sum_\beta (G_n)_{\alpha\beta} |L_\beta^{(n)}\rangle$. Now we may choose the transformation G_n, such that in the new gauge, these states are orthonormal $\langle \tilde{L}_\alpha^{(n)} | \tilde{L}_{\alpha'}^{(n)} \rangle = \delta_{\alpha,\alpha'}$. If we iterate this gauge choice for all bonds from left to right, we find a new set of tensors $A^{[1]}, \dots, A^{[N]}$ which give the same state $|\mathrm{MPS}(M^{[1]}, \dots, M^{[N]}))\rangle = \mathcal{N} |\mathrm{MPS}(A^{[1]}, \dots, A^{[N]}))\rangle$, possibly up to a normalization factor $\mathcal{N} > 0$, and fulfill the (left) isometric property



(2.16)

Note that to enforce this property for the last tensor, we effectively just rescale it such that the MPS is normalized. We call such tensors “left isometric”, or “in (isometric) A form”, as they are left isometries when reshaped to a matrix $A^{[n]}_{(\alpha i),\beta}$.

To bring a given MPS to isometric A form in practice, note that we do not need to compute the transformation G_n. We only need to find the new tensors $A^{[n]}, \tilde{M}^{[n+1]}$ such that the former is in A form and such that the state is unchanged $A^{[n]} \cdot \tilde{M}^{[n+1]} = M^{[n]} \cdot M^{[n+1]}$. This can be achieved with a QR decomposition

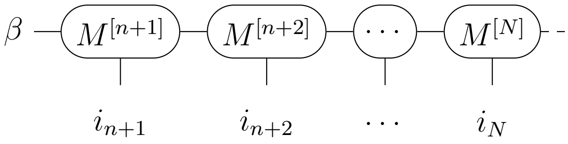


(2.17)

which for a full A form of the entire MPS needs to be iterated for all bonds, that is for $n = 1, \dots, N-1$. This procedure is contained as a special case $m = N$ in algorithm 2.1. Note that we only require the orthogonal property of the Q factor and do not use the triangular properties of the R factor in the QR decomposition.

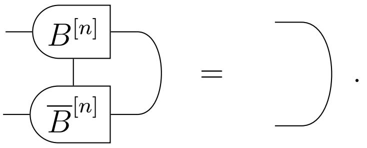
We can alternatively go right to left instead and map the family

$$\left|R^{(n)}_{\beta}\right\rangle = \sum_{i_{n+1},\dots,i_N}$$



$$|i_{n+1}, \dots, i_N\rangle \tag{2.18}$$

of states on the right of a given bond to an orthonormal set. Note that the superscript refers to the n-th *bond* of the system on which the index β lives, such that the first site of the state is site $n+1$, to the right of that bond. This yields MPS tensors in right isometric form, or isometric B form, with the property

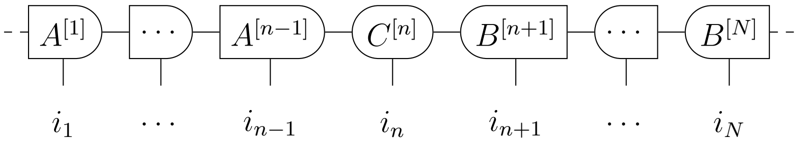


(2.19)

It can be achieved in practice by sweeping LQ decompositions from left to right, forming $X = M^{[n]} \cdot L$ and LQ decomposing $X = L \cdot B^{[n]}$ instead, see algorithm 2.1 with $m = 1$.

Or we can go to a mixed isometric form, where we gauge all bonds to the left of a given tensor n to A form and all bonds to the right to B form. As a result, we find

$$|\psi\rangle = \sum_{i_1,\dots,i_N}$$

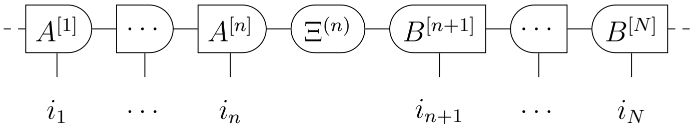


$$|i_1, \dots, i_N\rangle\,. \tag{2.20}$$

We call a tensor in isometric central form, or isometric C form, if the MPS can be written with only A tensors to its left and only B tensors to its right, as above. Such tensors are also referred to as an orthogonality center. It is common to normalize the C form tensor to $\|C^{[n]}\|_{\mathrm{F}} = 1$, as this gives a normalized MPS $\||\psi\rangle\| = 1$. A procedure to establish a mixed isometric form is given in algorithm 2.1.

We can perform an additional orthogonalization, e.g. a QR factorization of $C^{[n]}$, to shift the orthogonality center from a site to a bond and find

$$|\psi\rangle = \sum_{i_1,\dots,i_N}$$



$$|i_1, \dots, i_N\rangle\,. \tag{2.21}$$

Algorithm 2.1 Establishing an isometric form

Given an MPS $|\psi\rangle = |\mathrm{MPS}(M^{[1]}, \ldots, M^{[N]})\rangle$ and integer $m \in \{1, \ldots, N\}$, find tensors $A^{[1]}, \ldots, A^{[m-1]}$ in isometric A form, $C^{[m]}$ and tensors $B^{[m+1]}, \ldots, B^{[N]}$ in isometric B form such that $|\psi\rangle = \mathcal{N}|\mathrm{MPS}(A^{[1]}, \ldots, A^{[m-1]}, C^{[m]}, B^{[m+1]}, \ldots, B^{[N]})\rangle$, where $\mathcal{N} = \||\psi\rangle\|$. The result is in isometric B form if $m = 1$, in A form if $m = N$ and in a mixed isometric form otherwise. We employ the dot product notation (2.13).

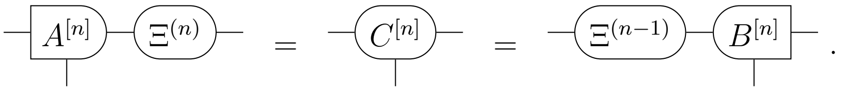
```
Initialize $R = 1 \in \mathbb{C}^{1\times 1}$.
For $n = 1, \ldots, m-1$ (left to right):
    Form $X = R \cdot M^{[n]}$
    Update $R$ and set $A^{[n]}$ by computing the QR factorization $X = A^{[n]} \cdot R$
Set $\tilde{M}^{[m]} = R \cdot M^{[m]}$
Initialize $L = 1 \in \mathbb{C}^{1\times 1}$.
For $n = N, N-1, \ldots, m+1$ (right to left):
    Form $X = M^{[n]} \cdot L$.
    Update $L$ and set $B^{[n]}$ by computing the LQ factorization $X = L \cdot B^{[m]}$.
Update $\tilde{M}^{[m]} \leftarrow \tilde{M}^{[m]} \cdot L$
Compute $\mathcal{N} = \|\tilde{M}^{[m]}\|$ and set $C^{[m]} = \tilde{M}^{[m]}/\mathcal{N}$.
```

If the given state $|\psi\rangle$ can be written in this fashion, with isometric A form tensor to the left of the bond and B tensors to its right, we call $\Xi^{(n)}$ a *bond matrix*. Note that in an isometric form, unlike in the stronger canonical form, the bond matrices are not determined by the state $|\psi\rangle$ alone but also depend on the choice of the A and B tensors. From this bond central form, we find

$$|\psi\rangle = \sum_{\alpha\beta} \Xi^{(n)}_{\alpha\beta} \left|L^{(n)}_\alpha\right\rangle \otimes \left|R^{(n)}_\beta\right\rangle . \tag{2.22}$$

This already looks structurally similar to the Schmidt decomposition (2.3), but we have not yet chosen the particular bases for the left/right subsystem in which the coefficients $\Xi^{(n)}_{\alpha\beta}$ become diagonal.

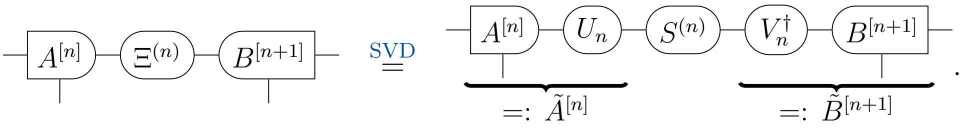
The A, B, and C isometric forms are related as follows



(2.23)

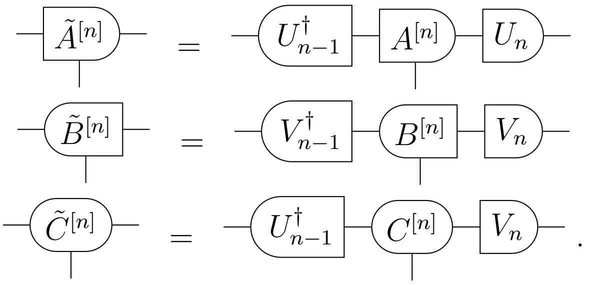
Note that an A tensor at the very right site is by definition also in C form $A^{[N]} = C^{[N]}$, and similarly, a B tensor at the very left $B^{[1]} = C^{[1]}$. This motivates the definition for the bond matrices $\Xi^{(0)} = \Xi^{(N)} = 1 \in \mathbb{C}^{1\times 1}$ for the trivial bonds at the system boundary, such that both sides of (2.23) hold also for $n = 1$ and $n = N$. Storing the bond matrices in addition to the MPS tensors allows convenient conversion between different forms on the fly. It does, however, require us to consider if the bond matrices remain valid when updating MPS tensors. Note that algorithm 2.1 does not yield the bond matrices. To establish an isometric form with all bond matrices, we instead need to use algorithm 2.2, but can relax the SVD to a deformed singular value decomposition (dSVD), that is a SVD-like decomposition $X = U \cdot \Xi^{(n)} \cdot B^{[n]}$ with a non-diagonal central factor $\Xi^{(n)}$, see section 3.1.4.

Demanding an isometric form (A, B or mixed) only partially fixes the gauge freedom, and transformations of the form (2.14) leave both the state and the isometric properties invariant, if we restrict to $G_n \in \mathrm{U}(\chi_n)$, i.e. to unitary gauge transformations. We can further fix some of the remaining gauge freedom by demanding the canonical form. We can take two equivalent perspectives on how to arrive there. First, instead of choosing G_n to map the $|L_\alpha^{(n)}\rangle$ to *any* orthonormal set, we may choose it to map to the particular orthonormal set given by the left Schmidt states. This gives rise to the procedure summarized in algorithm 2.2, which is similar to algorithm 2.1, except we need to perform an SVD instead of QR decompositions. Alternatively, we can gauge an existing isometric form with bond matrix (2.21) to match the Schmidt decomposition by performing an SVD of the bond matrix



(2.24)

Thus, if we have access to all bond matrices and perform their SVD $\Xi^{(n)} =: U_n S^{(n)} V_n^\dagger$, we obtain the Schmidt values $S^{(n)}$ for all bonds (which are the bond matrices of the canonical form) and can obtain the MPS tensors in canonical form (with tilde) from their counterparts in isometric form (no tilde) as follows



(2.25)

As an explicit definition, an MPS is in left canonical form, or canonical A form, if the left states $|L_\alpha^{(n)}\rangle$, defined in (2.15) are left Schmidt states of the bipartition on every bond n. Similarly it is in right canonical form, or B form, if the right states $|R_\beta^{(n)}\rangle$ in (2.18) are right Schmidt states of the bipartition on every bond n, and in mixed canonical form if the A property holds up to some bond, while the B property holds for the rest. Thus, a canonical form is a special case of an isometric form, where the bond matrices $\Xi^{(n)}$ are diagonal, real, and non-negative, which means that they are the Schmidt values $S^{(n)}$ of $|\psi\rangle$ w.r.t. a bipartition on bond n. The canonical form almost fixes the gauge; the remaining freedom is $G_n \in \bigoplus_{i=1}^{\chi} \mathrm{U}(1)$, meaning a phase choice for every Schmidt state, or possibly larger $\mathrm{U}(N)$ if there is a multiplet of N degenerate Schmidt values.

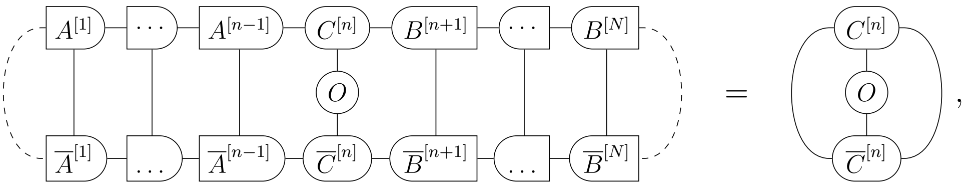
The main benefit of an isometric or canonical form are the isometric properties (2.16) and (2.19). They allow us to operate locally in algorithms. Firstly, expectation values of local operators, e.g. of $\langle\psi|O_n|\psi\rangle$ where O_n is a single site operator acting

Algorithm 2.2 Establishing a right canonical form

Given an MPS $|\psi\rangle = |\mathrm{MPS}(M^{[1]}, \dots, M^{[N]})\rangle$ and integer $n \in \{1, \dots, N\}$, find the Schmidt values $S^{(1)}, \dots, S^{(N-1)}$ of $|\psi\rangle$, as well as tensors $B^{[1]}, \dots, B^{[n-1]}$ in canonical B form, $C^{[n]}$, and tensors $A^{[n+1]}, \dots, A^{[N]}$ in canonical A form, such that $|\psi\rangle = \mathcal{N}|\mathrm{MPS}(B^{[1]}, \dots, B^{[N]})\rangle$, where $\mathcal{N} = \||\psi\rangle\|$. The result is in canonical B form if $n = 1$, in A form if $n = N$, and in mixed canonical form otherwise. We employ the dot product notation (2.13).

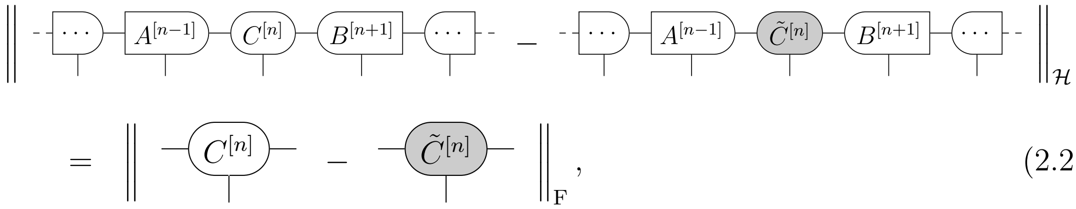
```
Initialize $W = 1 \in \mathbb{C}^{1\times 1}$ and use the dummy value $S^{(0)} = 1 \in \mathbb{C}^{1\times 1}$
For $m = 1, \dots, n-1$ (left to right):
    Form $X = S^{(m-1)} \cdot W \cdot M^{[m]}$.
    Update $W$ and set $A^{[m]}, S^{(m)}$ by computing the SVD $X = A^{[m]} \cdot S^{(m)} \cdot W$.
Set $\tilde{M}^{[n]} = S^{(n-1)} \cdot W \cdot M^{[n]}$.
Initialize $U = 1 \in \mathbb{C}^{1\times 1}$ and use the dummy value $S^{(N)} = 1 \in \mathbb{C}^{1\times 1}$.
For $m = N, \dots, n+1$ (right to left):
    Form $X = M^{[m]} \cdot U \cdot S^{(m)}$
    Update $U$ and set $S^{(m)}, B^{[m]}$ by computing the SVD $X = U \cdot S^{(m)} \cdot B^{[m]}$
Update $\tilde{M}^{[n]} \leftarrow \tilde{M}^{[n]} \cdot U \cdot S^{(n)}$.
Compute $\mathcal{N} = \|\tilde{M}^{[n]}\|$ and set $C^{[n]} = \tilde{M}^{[n]}/\mathcal{N}$.
```

on site n reduce to



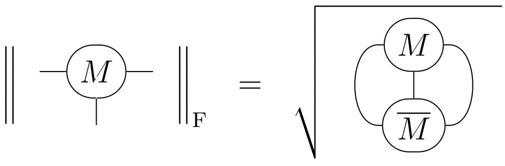
(2.26)

where we can obtain $C^{[n]}$ on the fly from any canonical form via (2.23), if we have access to the bond matrices. Secondly, the isometric property connects the norm $\|-\|_{\mathcal{H}}$ of the many-body Hilbert space to the local 2-norm $\|-\|_{\mathrm{F}}$ of single tensor(s) in the following sense. Consider modifying an MPS in mixed canonical form by changing the tensor at the orthogonality center. Then, the distance between these two states in the Hilbert space norm is simply given by the 2-norm distance of the respective tensors



(2.27)

where



$$= \sqrt{\sum_{\alpha,i,\beta} |M_{\alpha,i,\beta}|^2} \qquad (2.28)$$

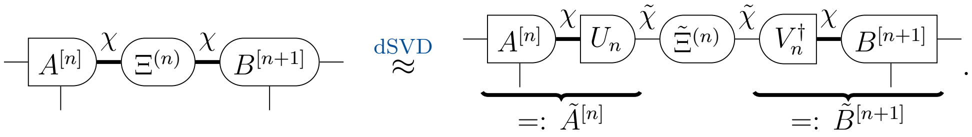
is the Frobenius norm of a tensor M.

As a result, we can derive updates locally, meaning such that they are optimal in the local norm $\|-\|_{\mathrm{F}}$ and get global optimality in the Hilbert space norm as a result. Having this feature without any caveats is unique to loop-free tensor networks such as MPS, and we believe it to be a main factor in the success of MPS algorithms.

The full canonical form, compared to an isometric form, has comparatively few additional benefits. For one, the canonical form allows direct access to the Schmidt values of the state, e.g. to extract bipartite entanglement entropies, which would require additional SVDs of the bond matrices in an isometric form. On the other hand, (almost) fixing the gauge may be beneficial for stability in some algorithmic settings.

The canonical form allows us to *conceptually* connect the MPS representation to a Schmidt decomposition with respect to any single bond. In particular, truncating that bond by keeping only some of the singular values and corresponding slices of MPS to either side of the bond realizes a truncation of the Schmidt decomposition and inherits the optimality properties of its error (2.6). Repeating this truncation on every bond yields an MPS approximation of the original state, and the resulting error can be controlled by tight bounds – see, e.g., Lemma 1 in [88]. In particular, we find that the required bond dimension to approximate an area law state up to some target error threshold is finite and independent of system size.

However, achieving the truncation in practice does not rely on the canonical form, as instead of truncating the singular values in a canonical form, we may employ any approximate low-rank factorization of the bond matrix in an isometric form. Thus, to truncate a single bond of an isometric MPS from dimension χ to $\tilde{\chi} < \chi$, we may perform an approximate factorization of the bond matrix as follows



. (2.29)

Here, the approximate factorization of $\Xi^{(n)}$ must have isometric factors $U_n, V_n^\dagger$ like an SVD, but the central factor $\tilde{\Xi}^{(n)}$ does not need to be diagonal. We call such a factorization a deformed singular value decomposition (dSVD), see section 3.1.4. If a standard SVD, a special case of the above, is used, we end up in a canonical form for this bond, but in general, the result is only in isometric form. The local truncation error of this approximate factorization is equal to the global error we incur w.r.t. the original many-body state because of the isometric properties. Truncating all bonds in a general isometric form (not requiring the bond to be an orthogonality center) is achieved by approximately factorizing all bond matrices $\Xi^{(n)} \approx: U_n \tilde{\Xi}^{(n)} V_n^\dagger$ and then updating the MPS analogous to (2.25).

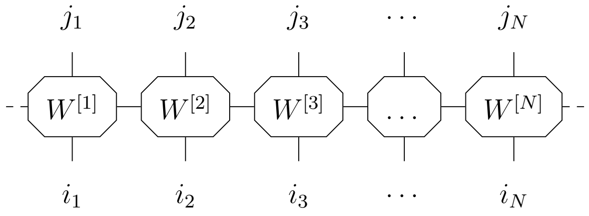
Of course, if the MPS was in canonical form to begin with – i.e. such that the bond matrix $\Xi^{(n)} = S^{(n)}$ is diagonal, real, positive – the low-rank approximation can be read off directly, by setting $U_n^\dagger = V_n^\dagger$ to the $\tilde{\chi} \times \chi$ projector that keeps the largest singular values, i.e. such that $\tilde{S}^{(n)} = U_n^\dagger S^{(n)} V_n$ is diagonal and contains the largest $\tilde{\chi}$ singular values. In that case, the truncation can be carried out cheaply by extracting rows/columns.

Note that while truncating a single bond in this fashion gives the optimal error for the given bond dimension (if using an optimal local truncation), a sequence of multiple such truncation steps that is required to truncate all bonds of an MPS to below some threshold is in general not optimal. We discuss an improved algorithm based on variational updates in section 2.3.5.

2.3.2 Matrix Product Operators (MPOs)

Several MPS algorithms, e.g. DMRG, TDVP and MPO evolution, rely on a compatible representation of the relevant operators, e.g. the Hamiltonian or evolution operator, in the form of a matrix product operator (MPO). An MPO is an operator whose coefficients in the computational basis are given in the form of a tensor network with a structure similar to an MPS, i.e. an operator of the form

$$\sum_{\substack{i_1,\dots,i_N \\ j_1,\dots,j_N}}$$



$$|i_1\rangle\langle j_1| \otimes \cdots \otimes |i_N\rangle\langle j_N| \ . \tag{2.30}$$

In fact, we can understand MPOs as MPSs in the space $\mathcal{H}_{\mathrm{op}} = \mathrm{Hom}\,(\mathcal{H},\,\mathcal{H}) \cong \mathcal{H} \otimes \mathcal{H}^\star$ of operators. We use a chamfered box for the MPO tensors, which has no particular meaning other than being a reminder that they parametrize an operator, not a state.

While in principle, any operator can be written as an MPO, the required bond dimension for general operators is exponential in system size. Hamiltonians of physical models, however, are commonly given in a highly structured form that allows explicit construction [89–92] of MPOs using finite state machines. This framework can be extended [93] to approximate the time evolution operator $\mathrm{e}^{-\mathrm{i}H\,\delta t}$ induced by such a Hamiltonian H in a Suzuki Trotter-like approximation assuming small δt. For complicated models, in particular models which are (a) two-dimensional, (b) have many degrees of freedom per site, and/or (c) have long-range couplings, the resulting bond dimensions may still be unfeasibly large. Since MPOs are MPSs in the doubled Hilbert space, they may be compressed, i.e. approximated by a lower bond dimension MPO, using MPS compression methods, as discussed in section 2.3.1 and 2.3.5.

2.3.3 Time evolving block decimation (TEBD)

The time evolving block decimation (TEBD) algorithm, originally devised as a time evolution algorithm, can be generalized to apply sequences of arbitrary nearest neighbor operators to an MPS, and approximate the resulting state again as an MPS.

Let us first derive this gate sequence for the time evolution induced by a nearest neighbor Hamiltonian $H = \sum_{n=1}^{N-1} h_{n,n+1}$. We can employ a Suzuki Trotter decomposition to approximately factorize its time evolution operator as

$$U(\delta t) = \mathrm{e}^{\mathrm{i}H\delta t} = \underbrace{U_{\mathrm{e}}(\delta t)\; U_{\mathrm{o}}(\delta t)}_{=:U^{(1)}(\delta t)} + \mathrm{O}(\delta t^2) \tag{2.31}$$

or

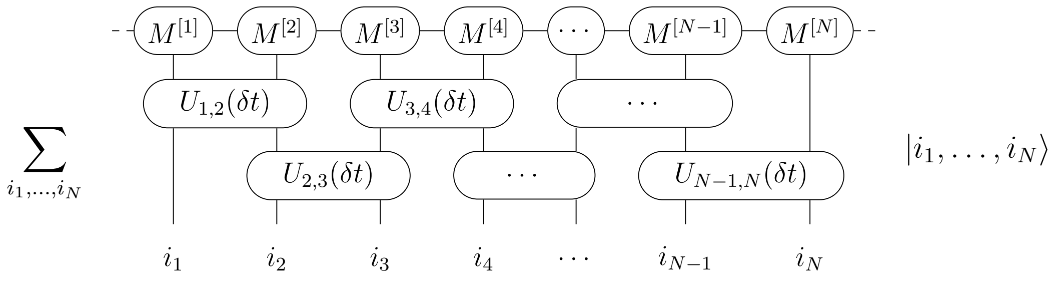
$$U(\delta t) = \underbrace{U_{\mathrm{e}}(\delta t/2)\; U_{\mathrm{o}}(\delta t)\; U_{\mathrm{e}}(\delta t/2)}_{=:U^{(2)}(\delta t)} + \mathrm{O}(\delta t^3) \tag{2.32}$$

in either first or second order, where

$$U_{\mathrm{e(o)}}(t) := \prod_{\substack{n=1\\ n \text{ even (odd)}}}^{N} U_{n,n+1}(t) \qquad ; \qquad U_{n,n+1}(t) := \mathrm{e}^{\mathrm{i}h_{n,n+1}t} \; . \tag{2.33}$$

This results in a brick-wall circuit for the time evolution operator. Given an initial MPS $|\psi_0\rangle = |\mathrm{MPS}(\{M^{[n]}\})\rangle$, the task for TEBD is then to approximate the action of e.g. a first order Trotterized step

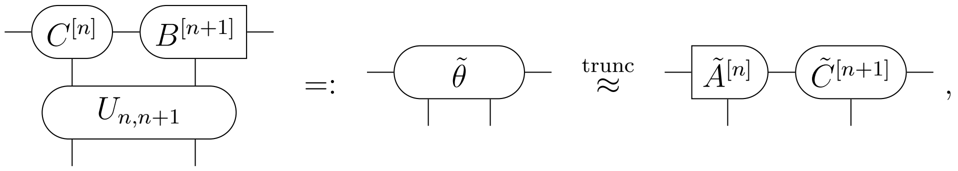
$U^{(1)}(\delta t)\,|\psi_0\rangle \quad =$



(2.34)

as an MPS with bounded bond dimension. Repeating this many times allows us to simulate time evolution by providing access to $[U^{(i)}(\delta t)]^m|\psi_0\rangle \approx [U(\delta t)]^m|\psi_0\rangle = |\psi(t = m\;\delta t)\rangle$.

The gates $U_{n,n+1}$ are then applied one after the other by updating only the two MPS tensors at the sites $n, n+1$ where the gate acts. The gate can be applied exactly, resulting in a two-site wavefunction $\tilde{\theta}$ given by

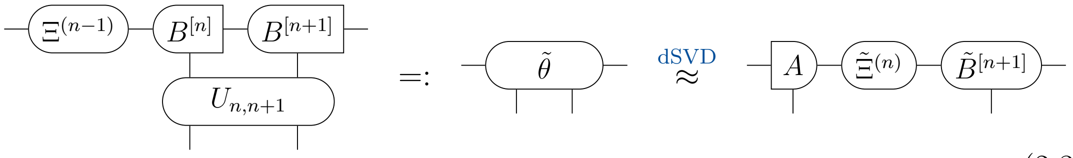


(2.35)

which subsequently needs to be factorized to retain the MPS form. To keep the bond dimension of the MPS bounded, this factorization needs to be truncated, e.g. using a truncated SVD $\tilde{\theta} \approx \tilde{A}^{[n]} \cdot (S \cdot W^{[n+1]}) =: \tilde{A}^{[n]} \cdot \tilde{C}^{[n+1]}$. It is crucial that the isometric form of the MPS is such that one of the tensors above the gate – here $C^{[n]}$ – is the orthogonality center. This guarantees that a locally optimal truncation of $\tilde{\theta}$ is optimal globally, analogous to (2.27).

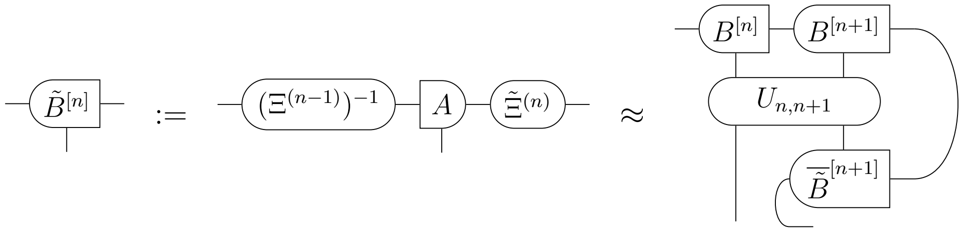
If the gates are unitary, this can be done by always maintaining a right isometric form[2] of the tensors, and keeping track of the bond matrices, to obtain local wavefuntions on the fly via (2.23), as suggested in [94]. This is because a unitary gate acting on sites $n, n+1$, applied according to the first equality in (2.35), preserves the defining property of the isometric B form for all other bonds. To the right of the update, this is trivial, since the right Schmidt-like states (2.18), that is $|R^{(m)}_\beta\rangle$ for $m > n$ are unchanged, while to the left, the Schmidt-like states $|R^{(m)}_\beta\rangle$ for $m < n$ retain their orthonormality if the gate is unitary. Since similar arguments apply to A form tensors, the bond matrices $\Xi^{(m)}$ for $m \neq n$, for all bonds except where the gate was applied, remain valid bond matrices. If the isometric form happens to be a canonical form, we can understand this from a different perspective: a unitary transformation acting only on one of the subsystems does not change the Schmidt values or states of a given bipartition, and thus, the canonical form is preserved for all bonds that the gate does not act on.

We can perform the TEBD update as



(2.36)

and find the new B form tensor for site $n+1$, as well as the new bond matrix for the updated MPS. From a deformed singular value decomposition (dSVD), we get the left tensor in an isometric A form, however. To restore the B form, we need to solve (2.23) and find



, (2.37)

where the approximate equality follows by left-multiplying (2.36) with $(\Xi^{(n-1)})^{-1}$ and projecting onto $\overline{\tilde{B}}^{[n+1]}$. It is the preferred way to form $\tilde{B}^{[n]}$ in practice since it avoids instabilities from inverting the bond matrix. Note that this uses the assumption of unitarity since we use the unmodified bond matrix $\Xi^{(n-1)}$ for bond $i-1$. Note also that the contraction for $\tilde{B}^{[n]}$ shares intermediate objects with the contraction of $\tilde{\theta}$. As a result, any sequence of gates can be applied in this fashion, and we

[2] This is a conventional choice, and operating in a left isometric form is possible too.

do not rely on the particular brickwall structure of the circuit that arises from the Trotterization.

If the gates are not unitary, on the other hand, applying the gate invalidates the bond matrices on *every* bond. This is because the orthonormality of the Schmidt-like states $|L_\alpha^{(n)}\rangle$ to its right or of the $|R_\beta^{(n)}\rangle$ to its left is broken by applying a gate, which invalidates any isometric A form to its right and B form to its left and thus invalidates all bond matrices. In particular, we may not use the unmodified $\Xi^{(n-1)}$ to convert the A form tensor that we get out of a deformed singular value decomposition (dSVD) to B form. Additionally, the isometric or canonical form of the MPS tensors is only preserved if it is a mixed canonical form with orthogonality center at either of the two active sites $n, n+1$. Thus, in this case, the update needs to be performed according to equation (2.35). Since the bond matrices are immediately invalidated when the next gate in a sequence is applied, there is no need to store them in non-unitary TEBD.

There are two ways of carrying out the truncated factorization, which is a truncated QR-like decomposition (tQR) as discussed in section 3.1.3. We can either have $\tilde{\theta} \approx \tilde{A}^{[n]} \cdot \tilde{C}^{[n+1]}$, which has the isometric factor on the left and is suitable for a subsequent update to the right, on sites $n+1, n+2$. Alternatively, we can have $\tilde{\theta} \approx \tilde{C}^{[n]} \cdot \tilde{B}^{[n+1]}$ with a right isometry $\tilde{B}^{[n+1]}$ which is suitable for updating the bond to the left next. Thus, a staircase pattern is more convenient for non-unitary TEBD than a brickwall circuit. If the brickwall structure is unavoidable, we may intersperse steps that simply shift the orthogonality center, using e.g. a QR decomposition for an isometric form or an SVD if a full canonical form is enforced, similar to (2.51).

The computational cost of TEBD is typically given as $\mathrm{O}(d^3\chi^3)$, where the dominant contribution comes from the factorization of $\tilde{\theta}$. Using a standard truncated SVD for this step results in the above scaling. However, using cheaper truncated factorization routines, we can bring the cost down to $\mathrm{O}(d^2\chi^3)$, as we discuss in chapter 3. The cost can not be brought down further than that by improving the factorization, as the cost of forming $\tilde{\theta}$ from the MPS tensors is in $\mathrm{O}(d^2\chi^3)$.

The TEBD algorithm is limited to nearest neighbor gates, and thus can only directly simulate time evolution of a nearest neighbor Hamiltonian. Models with slightly longer-ranged couplings (or circuits with few body gates) can be addressed by grouping sites until the couplings between effective sites are nearest neighbor terms again. This comes at the cost of increasing the dimension d of the effective local Hilbert space – and thus also the cost of all algorithms – exponentially and is therefore limited to very short-ranged couplings. For general longer range terms, time evolution can be simulated using MPO evolution, see section 2.3.5, or TDVP [85, 86].

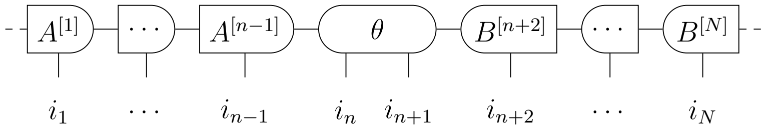
The TEBD algorithm can also be used to approximate the *imaginary* time evolution of a given nearest neighbor Hamiltonian, which projects the initial state onto the ground state in the limit of infinite evolved imaginary time. Thus, imaginary time TEBD can, in principle, be used as a ground state search. This is, however, suboptimal, as such a method is restricted to nearest neighbor models (after potentially grouping sites), slow to converge, and prone to local minima. In virtually all cases, the DMRG method gives better results, faster.

2.3.4 Density matrix renormalization group (DMRG)

Due to its long history, and reformulation as a tensor network method, there are several ways to understand why the DMRG algorithm for MPS ground state search works. Here, we choose the perspective of a variational updating algorithm, where the goal is minimizing the energy $E = \langle\psi|H|\psi\rangle$ of a trial MPS $|\psi\rangle$ while keeping $\langle\psi|\psi\rangle = 1$. We assume the Hamiltonian $H = \mathrm{MPO}(W^{[1]}, \dots, W^{[N]})$ is given as an MPO. The strategy is then to update the MPS tensors on two neighboring sites, the "active sites", while keeping the other tensors fixed. Starting from some initial guess in MPS form, these updates are then repeated for all pairs of sites, and for a number of iterations, until e.g. the energy of the trial state converges. Since the update is, in general, not unitary, the discussion regarding TEBD with non-unitary gates applies, and an isometric form of the MPS is only preserved by the update if the form has an orthogonality center at either one of the active sites. Moreover, bond matrices Ξ for the other bonds are invalidated by the update. Therefore, the isometric / canonical form is handled such that we always have A tensors to the left of the active site(s) and B tensors to the right, and there is no need to keep track of the bond matrices.

To derive the update, we first find the optimal two-site wavefunction $\theta_{\alpha,i,j,\beta}$ such that the modified MPS-like state

$$|\psi\rangle = \sum_{i_1,\dots,i_N}$$



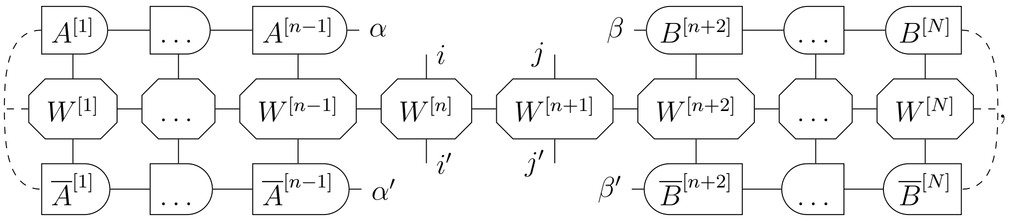
$$|i_1, \dots, i_N\rangle \tag{2.38}$$

gives the lowest possible energy, where $A^{[1]}, \dots, A^{[n-1]}$ and $B^{[n+2]}, \dots, B^{[N]}$ are the MPS tensors of the current best guess, before the update. The optimization problem, formulated in terms of θ is

$$\theta^\dagger \cdot H_{\text{eff}}^{n,n+1} \cdot \theta \to \text{MIN} \qquad \text{while} \quad \theta^\dagger \cdot N_{\text{eff}}^{n,n+1} \cdot \theta = 1 \ , \tag{2.39}$$

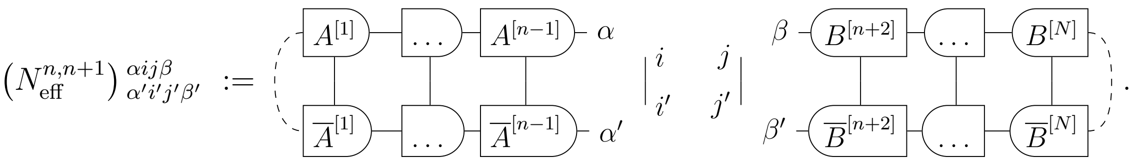
where the effective Hamiltonian $H_{\text{eff}}^{n,n+1}$ and effective norm matrix $N_{\text{eff}}^{n,n+1}$ for active sites $(n, n+1)$ consist of the other tensors and the MPO and are in particular constant as a function of θ. They are given by

$$\left(H_{\text{eff}}^{n,n+1}\right)_{\alpha' i' j' \beta'}^{\alpha i j \beta} \quad :=$$



(2.40)

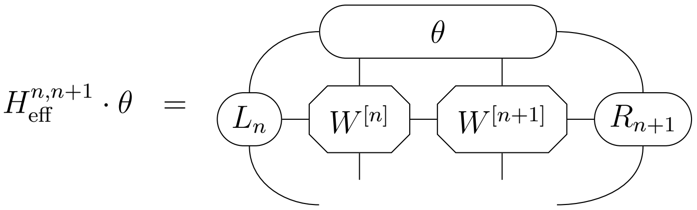
where $\{W^{[n]}\}$ are the MPO tensors of the Hamiltonian, as well as



(2.41)

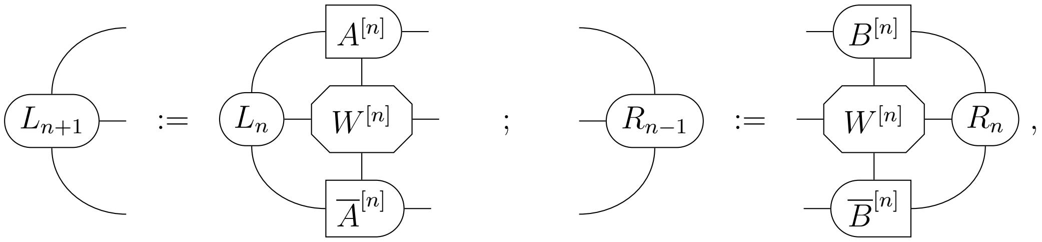
The effective norm matrix $N_{\text{eff}}^{n,n+1} = \mathbb{1}$ reduces to the identity because of the isometric properties (2.16) and (2.19). Therefore, we can find the optimal two-site update by finding the lowest eigenvector of the eigenvalue problem $H_{\text{eff}}^{n,n+1} \cdot \theta = \epsilon\theta$ and directly obtain the energy ϵ of the resulting state. Conceptually, we should find a normalized eigenvector, such that $\theta^\dagger\theta = 1$, but this can be taken care of by the normalization after the truncation step in practice.

The effective Hamiltonian is given in a factorized form, such that matrix-vector products



(2.42)

can be computed more efficiently than forming the whole H_{eff} matrix. The left and right environment tensors are defined recursively as



(2.43)

with the base cases $L_1 = R_N = 1 \in \mathbb{C}^{1\times1\times1}$. Additionally, we have access to an initial guess for the solution – the two site wavefunction from the MPS before the update – which becomes better and better the closer the outer DMRG loop is to convergence. For these two reasons, it is beneficial to use an iterative eigensolver, such as e.g. the Lanczos algorithm, to find the ground state $\tilde{\theta}$ of $H_{\text{eff}}^{n,n+1}$.

At this point, we proceed similar to the TEBD algorithm to renormalize the two-site update $\tilde{\theta}$, by approximating it in MPS form with a bounded bond dimension. This two-site update is then repeated for all pairs of sites, sweeping first left to right, then right to left. These sweeps are repeated until a convergence condition is met. Typically, one looks at the convergence of energy and bipartite entanglement entropy. If both are no longer changing substantially, with relative changes below a threshold of e.g. $\sim 10^{-8}$, we consider the algorithm converged. The resulting procedure is summarized in algorithm 2.3. We call it "simple" since additional algorithmic considerations, such as dynamically increasing the maximum bond dimension or adding

Algorithm 2.3 Simple two-site finite DMRG

Given a hermitian operator $H = \mathrm{MPO}(W^{[1]}, \ldots, W^{[N]}))$, a maximum bond dimension χ, and an initial guess $|\psi\rangle = |\mathrm{MPS}(\tilde{M}^{[1]}, \ldots, \tilde{M}^{[N]})\rangle$, computes a normalized MPS approximation $|\phi\rangle = |\mathrm{MPS}(M^{[1]}, \ldots, M^{[N]})\rangle$ of the ground state of H with bond dimension at most χ. Regarding notation, we think of only a single variable $M^{[n]}$ for every site that stores the MPS tensor for the current best guess. We use subscripts to keep track of its isometric form, but e.g. $M_A^{[n]}$ and $M_B^{[n]}$ refer to the same variable. We employ the dot product notation (2.13).

```
Initialize MPS tensors $M_A^{[1]}, M_C^{[2]}, M_B^{[3]}, \ldots, M_B^{[N]}$ by bringing $|\psi\rangle$ to a mixed isometric form with orthogonality center on site $m = 2$, e.g. using algorithm 2.1.
Compute the right environments $R_n$ for $n = N, \ldots, 2$ using (2.43).
Initialize the left environments $L_n$ for $n = 1, \ldots, N-1$ with placeholders.
Repeat until convergence:
    For $n = 1, \ldots, N-2$ (right to left):
        Form $\theta_0 = M^{[n]} \cdot M^{[n+1]}$ (the isometric form is $M_A^{[1]} M_C^{[2]}$ or $M_C^{[n]} M_B^{[n+1]}$).
        Find the ground state $\tilde{\theta}$ of $H_{\text{eff}}^{n,n+1}$, (2.42), iteratively with initial guess $\theta_0$.
        Decompose $\tilde{\theta} \approx U^{[n]} \cdot C^{[n+1]}$ with rank $\leq \chi$ such that $U^{[n]}$ is left isometric.
        Update $M_A^{[n]} \leftarrow U^{[n]}$ and $M_C^{[n+1]} \leftarrow C^{[n+1]} / \|C^{[n+1]}\|$.
        Update $L_{n+1}$ using (2.43). Note that the $R_n$ is now invalid.
    For $n = N-1, \ldots, 2$ (left to right):
        Form $\theta_0 = M^{[n]} \cdot M^{[n+1]}$ (the isometric form is $M_C^{[N-1]} M_B^{[N]}$ or $M_A^{[n]} M_C^{[n+1]}$).
        Find the ground state $\tilde{\theta}$ of $H_{\text{eff}}^{n,n+1}$, (2.42), iteratively with initial guess $\theta_0$.
        Decompose $\tilde{\theta} \approx C^{[n]} \cdot W^{[n+1]}$ at rank $\leq \chi$ s.t. $W^{[n+1]}$ is right isometric.
        Update $M_C^{[n]} \leftarrow C^{[n]} / \|C^{[n]}\|$ and $M_B^{[n+1]} \leftarrow W^{[n+1]}$
        Update $R_n$ using (2.43). Note that $L_{n+1}$ is now invalid.
```

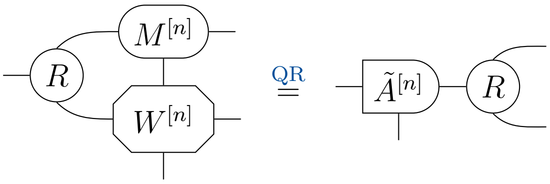
density matrix perturbations [57] are omitted. The decomposition in steps 8 and 14 can e.g. be performed using truncated SVDs, or with other approximate low rank decompositions, see section 3.1. For completeness, let us mention that single-site versions of DMRG have been formulated [56], which we do not discuss here.

The computational cost of the two-site algorithm is in $\mathrm{O}(d^2\eta\chi^3)$, where d is the dimension of the local Hilbert space, η the bond dimension of the MPO and χ the MPS bond dimension, assuming $d\eta \leq \chi$. Under the same assumptions, the pure[3] single-site version has a cost in $\mathrm{O}(d\eta\chi^3)$.

2.3.5 MPO Time evolution

Let us now discuss methods to approximately apply a matrix product operator (MPO) $O = \mathrm{MPO}(W^{[1]}, \dots, W^{[N]})$ to an MPS $|\psi\rangle = |\mathrm{MPS}(M^{[1]}, \dots, M^{[N]})\rangle$, that is to approximate $O|\psi\rangle \approx |\phi\rangle$ by a new MPS $|\phi\rangle = |\mathrm{MPS}(\tilde{M}^{[1]}, \dots, \tilde{M}^{[N]})\rangle$ *of bounded bond dimension*. This problem may arise from time evolution if $O = \mathrm{e}^{-\mathrm{i}Ht}$ is (an approximation of) the time evolution operator of a given system, but can also arise in different settings, such as simulating quantum circuits. In either case, it is known under the keyword "MPO evolution".

A straight-forward approach is to first perform the contraction exactly and then – if needed – truncate the bond dimension similar to (2.24), by establishing an isometric form in a first sweep of QR steps and then truncating using a second sweep of SVD steps. It turns out that the contraction and the QR sweep can be conveniently achieved in parallel by iterating



(2.44)

from left to right, such that $|\mathrm{MPS}(\tilde{A}^{[1]}, \dots, \tilde{A}^{[N]})\rangle = \frac{1}{\mathcal{N}} O|\psi\rangle$ equals the target state exactly (up to normalization) and is in left isometric form. The resulting MPS has bond dimension $\eta\chi$, where η is the bond dimension of the operator O and χ the bond dimension of the state $|\psi\rangle$. Next, we sweep right to left with truncated SVDs to achieve an approximation with the desired bond dimension. The resulting procedure is summarized in algorithm 2.4.

Here, the isometric form guarantees that if a single truncation is optimal locally, e.g. by using a truncated SVD, it is also optimal globally. However, the sequence of multiple truncations in the SVD-based compression scheme is not optimal and, in general, does not find the best MPS approximation at a given bond dimension. Better approximations can be obtained with a variational algorithm, similar to the DMRG algorithm, but instead of minimizing the energy, we minimize the square

[3] Some approaches to add density matrix perturbations ("mixing") have a higher cost scaling, namely in $\mathrm{O}(d^2\eta\chi^3)$, same as the two-site DMRG, though typically with a smaller prefactor. We do not consider these costs here.

Algorithm 2.4 SVD-based MPO compression

Given $O = \mathrm{MPO}(W^{[1]}, \dots, W^{[N]})$ and $|\psi\rangle = |\mathrm{MPS}(M^{[1]}, \dots, M^{[N]})\rangle$ and a bond dimension χ, find a normalized MPS approximation $O|\psi\rangle \approx \mathcal{N}|\mathrm{MPS}(\tilde{B}^{[1]}, \dots, \tilde{B}^{[N]})\rangle$ in right isometric form with bond dimension $\leq \chi$, where $\mathcal{N} > 0$ is the norm of the approximation. We employ the dot product notation (2.13).

```
 1. Initialize $R = 1 \in \mathbb{C}^{1\times 1\times 1}$
 2. For $n = 1, \dots, N-1$
 3.     Update $R$ and set $\tilde{A}^{[n]}$ via the contraction and QR decomposition of (2.44)
 4. Set $\tilde{C}^{[N]}$ as the LHS of (2.44), without decomposing it.
 5. Compute $\mathcal{N}_1 = \|\tilde{C}^{[N]}\|$ and normalize $\tilde{C}^{[N]} \leftarrow \tilde{C}^{[N]}/\mathcal{N}_1$.
 6. Initialize $C = 1 \in \mathbb{C}^{1\times 1}$.
 7. For $n = N, \dots, 2$:
 8.     Form $X = \tilde{A}^{[n]} \cdot C$.
 9.     Set $C$ and $\tilde{B}^{[n]}$ via truncated SVD $X \approx U \cdot (SW^{[n]}) =: C \cdot \tilde{B}^{[n]}$ at rank $\leq \chi$.
10. Compute $\mathcal{N}_2 = \|C\|$ and set $\tilde{B}^{[1]} = C/\mathcal{N}_2$ as well as $\mathcal{N} = \mathcal{N}_1\mathcal{N}_2$.
```

distance

$$\Delta^2 = \left\| |\phi\rangle - U\,|\psi\rangle \right\|^2 = \langle\phi|\phi\rangle - 2\mathrm{Re}\,\langle\phi|U|\psi\rangle + \text{const.} \tag{2.45}$$

and parametrize the trial state $|\phi\rangle = \mathcal{N}|\tilde{M}^{[1]}, \dots, \tilde{M}^{[N]}\rangle$ as a normalized MPS and a normalization factor $\mathcal{N} > 0$.

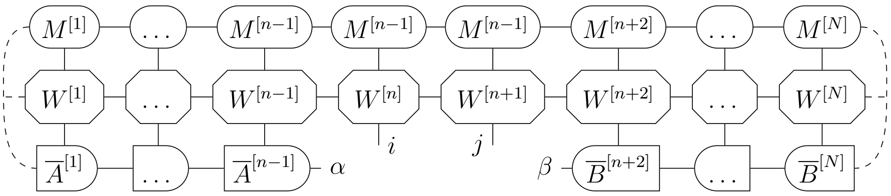
If we again focus on a two-site version first, the local problem to solve for a two-site wavefunction θ at an orthogonality center at sites $n, n+1$ is now

$$\Delta^2 = \theta^\dagger \cdot \theta - 2\mathrm{Re}\ \theta^\dagger \cdot \varphi_{\text{eff}}^{n,n+1} + \text{const.} \to \text{MIN}\ . \tag{2.46}$$

The overlap is taken with $\varphi_{\text{eff}}^{n,n+1}$, the evolved state projected by the other tensors of the trial state

$$\left(\varphi_{\text{eff}}^{n,n+1}\right)_{\alpha,i,j,\beta}$$

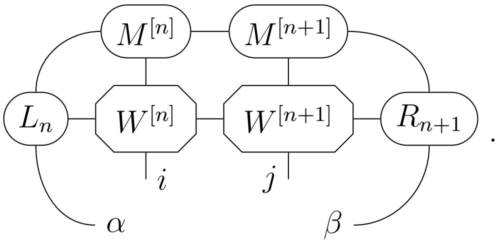
$:=$



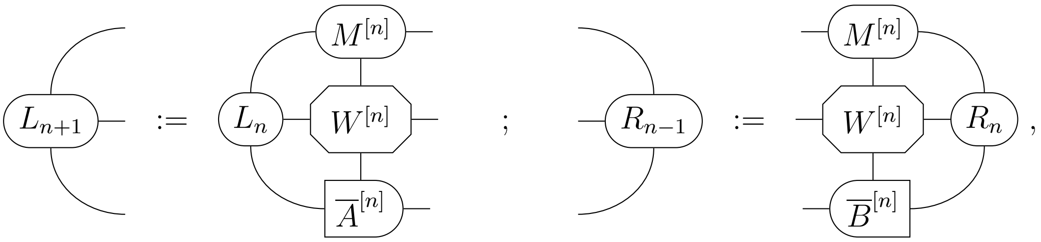
, (2.47)

where $M^{[n]}$ are the MPS tensors of the old state $|\psi\rangle$ and $A^{[n]}, B^{[n]}$ are the MPS tensors of the trial state that is currently updated. Unlike DMRG, where we have to solve for the ground state of the effective Hamiltonian numerically, this local problem has the closed form solution $\tilde{\theta} = \varphi_{\text{eff}}^{n,n+1}$. Like in DMRG, it is practical to contract it using environment tensors, which are partial contractions of (2.47) that can be reused for subsequent updates on other sites

$$\left(\varphi_{\text{eff}}^{n,n+1}\right)_{\alpha,i,j,\beta} \quad = \qquad \text{(2.48)}$$



Note that the environment tensors now have MPS tensors of the old state $|\psi\rangle$ in the ket layer, in any isometric form, and tensors of the current trial MPS in fixed isometric form in the bra layer



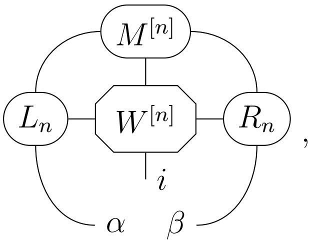
(2.49)

where again the base cases are $L_1 = 1 \in \mathbb{C}^{1\times1\times1} = R_N$.

Now, like in DMRG, after solving the local problem and finding the optimal two-site wavefunction $\tilde{\theta}$, we need to factorize it and truncate to restore the MPS form. Note that a subsequent update on a neighboring pair of sites immediately overrides one tensor from the previous update, such that it is not necessary to compute it, and we only need both factors at the very end of a sweep where the algorithm may terminate. A common convergence criterion for the outer loop is a convergence of the square distance Δ^2. The resulting procedure is summarized in algorithm 2.5.

Instead of updating two active sites at a time, a single-site version of the variational algorithm can be formulated. For a local update, we require the current trial MPS to be in mixed canonical form with orthogonality center at the active site. Analogously to the two-site version, the optimal single-site update is $\tilde{C}^{[n]} = \varphi_{\text{eff}}^n$, which is given by

$$\left(\varphi_{\text{eff}}^{n}\right)_{\alpha,i,\beta} \quad = \qquad \text{(2.50)}$$



using the same environments (2.49) as for the two-site version. There is no truncation step required, such that we need to shift the orthogonality center explicitly, e.g. using a QR decomposition on a right sweep

$$\tilde{C}^{[n]} \cdot B^{[n+1]} \overset{\text{QR}}{=} \tilde{A}^{[n]} \cdot R \cdot B^{[n+1]} =: \tilde{A}^{[n]} \cdot \tilde{C}^{[n+1]} \qquad \text{(2.51)}$$

Algorithm 2.5 Two-site variational MPO compression

Given an operator $O = \text{MPO}(W^{[1]}, \dots, W^{[N]}))$, a state $|\psi\rangle = |\text{MPS}(M^{[1]}, \dots, M^{[N]})\rangle$, a maximum bond dimension χ, and an initial guess $|\phi_0\rangle$ in MPS form, computes an MPS approximation $|\phi\rangle = \mathcal{N}|\text{MPS}(\tilde{M}^{[1]}, \dots, \tilde{M}^{[N]})\rangle \approx O|\psi\rangle$ with bond dimension at most χ. This is in terms of a normalized MPS and the factor $\mathcal{N} > 0$ is the norm of the approximation. We employ the same notation as in algorithm 2.3, where e.g. $\tilde{M}_A^{[n]}$ and $\tilde{M}_B^{[n]}$ refer to the same variable and the subscript only indicates its current isometric form. We employ the dot product notation (2.13).

```
Initialize MPS tensors $\tilde{M}_A^{[1]}, \tilde{M}_C^{[2]}, \tilde{M}_B^{[3]}, \dots, \tilde{M}_B^{[N]}$ by bringing $|\phi_0\rangle$ to a mixed isometric form with orthogonality center on site $m = 2$, e.g. using algorithm 2.1.
Compute the right environments $R_n$ for $n = N, \dots, 2$ using (2.49).
Initialize the left environments $L_n$ for $n = 1, \dots, N-1$ with placeholders.
Repeat until convergence:
    For $n = 1, \dots, N-2$: (right sweep)
        Form $\varphi_{\text{eff}}^{n,n+1}$ given by (2.48).
        Decompose $\varphi_{\text{eff}}^{n,n+1} \approx U^{[n]} \cdot C^{[n+1]}$ at rank $\leq \chi$ s.t. $U^{[n]}$ is left isometric.
        Update $\tilde{M}_A^{[n]} \leftarrow U^{[n]}$. Note that there is no need to update $\tilde{M}^{[n+1]}$.
        Update $L_{n+1}$ using (2.49). Note that the $R_n$ is now invalid.
    For $n = N-1, \dots, 2$ (left sweep)
        Form $\varphi_{\text{eff}}^{n,n+1}$ given by (2.48).
        Decompose $\varphi_{\text{eff}}^{n,n+1} \approx C^{[n]} \cdot W^{[n+1]}$ at rank $\leq \chi$ s.t. $W^{[n+1]}$ is right isometric.
        Update $\tilde{M}_B^{[n+1]} \leftarrow W^{[n+1]}$. Note that $\tilde{M}^{[n]}$ is not needed unless $n = 2$.
        Update $R_n$ using (2.49). Note that $L_{n+1}$ is now invalid.
    Compute $\mathcal{N} = \|C^{[2]}\|$ and update $\tilde{M}_C^{[2]} \leftarrow C^{[2]}/\mathcal{N}$.
```

and could then proceed to update the active next site $n + 1$. Note that if there is a subsequent update, i.e. unless the algorithm terminates, there is no need to compute $\widetilde{C}^{[n+1]}$, as it will be overridden in the next update anyway. The resulting procedure is summarized in algorithm 2.6.

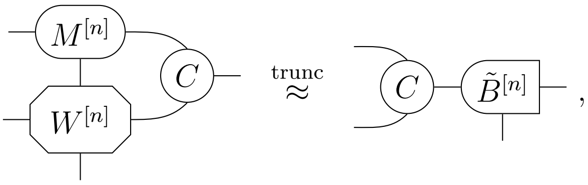
Algorithm 2.6 Single-site variational MPO compression

Given an operator $O = \text{MPO}(W^{[1]}, \ldots, W^{[N]})$, a state $|\psi\rangle = |\text{MPS}(M^{[1]}, \ldots, M^{[N]})\rangle$, and an initial guess $|\phi_0\rangle$ in MPS form, computes an MPS approximation $|\phi\rangle = \mathcal{N}|\text{MPS}(\tilde{M}^{[1]}, \ldots, \tilde{M}^{[N]})\rangle \approx O|\psi\rangle$ with the same bond dimensions as $|\phi_0\rangle$. This is in terms of a normalized MPS and the factor $\mathcal{N} > 0$ is the norm of the approximation. We employ the same notation as in algorithm 2.3, where e.g. $\tilde{M}_A^{[n]}$ and $\tilde{M}_B^{[n]}$ refer to the same variable and the subscript only indicates its current isometric form. We employ the dot product notation (2.13).

```
Initialize MPS tensors $\tilde{M}_C^{[1]}, \tilde{M}_B^{[2]}, \tilde{M}_B^{[3]}, \ldots, \tilde{M}_B^{[N]}$ by bringing $|\phi_0\rangle$ to a mixed
isometric form with orthogonality center on site $n = 1$, e.g. using algorithm 2.1.
Compute the right environments $R_n$ for $n = N, \ldots, 1$ using (2.49).
Initialize the left environments $L_n$ for $n = 1, \ldots, N$ with placeholders.
Repeat until convergence:
  For $n = 1, \ldots, N - 1$: (right sweep):
    Form $\varphi_{\text{eff}}^n$ given by (2.50).
    Update $\tilde{M}_A^{[n]}$ by computing the QR decomposition $\varphi_{\text{eff}}^n = \tilde{M}_A^{[n]} \cdot R$.
    Update $L_{n+1}$ using (2.49). Note that the $R_{n-1}$ is now invalid.
  For $n = N, \ldots, 2$ (left sweep):
    Form $\varphi_{\text{eff}}^n$ given by (2.50).
    Update $\tilde{M}_B^{[n]}$ by computing the LQ decomposition $\varphi_{\text{eff}}^n = L \cdot \tilde{M}_A^{[n]}$.
    Update $R_{n-1}$ using (2.49). Note that $L_{n+1}$ is now invalid.
  Update $\tilde{M}_C^{[1]} \leftarrow \tilde{M}_A^{[1]} \cdot L$.
  Compute $\mathcal{N} = \|\tilde{M}_C^{[1]}\|$ and update $\tilde{M}_C^{[1]} \leftarrow \tilde{M}_C^{[1]} / \mathcal{N}$.
Optionally, establish a full canonical form using algorithm (2.2).
```

Note that unlike for the two-site version, the bond dimension of the trial MPS can not dynamically grow and is fixed by the initial guess, which should therefore be chosen with the full target bond dimension. This can be remedied by incorporating subspace expansion methods ("mixing") similar to the adjustments in single-site DMRG [56, 57]. While this seems to be strictly necessary to get good results with single-site DMRG, single-site MPO evolution can work well even without mixing.

Both of these variational algorithms require an initial guess. In a general setting we may start from well-chosen states with exact MPS representations, such as product states, from an MPS with random tensor entries, or from a state that results from any other MPO evolution method, such as the SVD-based compression of algorithm 2.4. For this particular purpose, a modified version of the SVD-based compression may be employed, that trades accuracy for performance, which is commonly called the zip-up method. It is similar to the SVD-based method, but directly performs truncation, even if there is no isometric form established yet. This is done by iterating



(2.52)

with a truncated QR-like decomposition (tQR) factorization, or rather an LQ version of it, see section 3.1.3. This is clearly sub-optimal and in general the results are not accurate enough to use this as a stand-alone method, but it is significantly cheaper in the presence of strong truncation. We summarize this approach in algorithm 2.7.

Algorithm 2.7 Zip-up MPO compression

Given an operator $O = \mathrm{MPO}(W^{[1]}, \ldots, W^{[N]}))$, a state $|\psi\rangle = |\mathrm{MPS}(M^{[1]}, \ldots, M^{[N]}))$ and a maximum bond dimension χ, computes a *crude* approximation $|\phi\rangle = \mathcal{N}|\mathrm{MPS}(\tilde{B}^{[1]}, \ldots, \tilde{B}^{[N]}))\rangle \approx O|\psi\rangle$ with a normalized MPS in isometric B form, bond dimension at most χ and norm $\mathcal{N} > 0$. The resulting approximation is generally of low quality and intended as an initial guess only. We employ the dot product notation (2.13).

```
1. Initialize $C = 1 \in \mathbb{C}^{1\times 1}$
2. For $n = N, N-1, \ldots, 2$ (right to left):
3.    Update $C$ and set $\tilde{B}^{[n]}$ via the contraction and factorization of (2.52).
4. Set $\tilde{C}^{[1]}$ as the LHS of (2.52), without decomposing it.
5. Compute $\mathcal{N} = \|\tilde{C}^{[1]}\|$ and set $\tilde{B}^{[1]} = \tilde{C}^{[1]}/\mathcal{N}$
```

We give the computational cost for all four methods in table 2.1.

Method	Definition	Cost scaling	Dominant step(s)
Two-Site Var.	Alg. 2.5	$\mathrm{O}(d^2\eta\chi^3) + \mathcal{C}_{\mathrm{init}}$	Contractions for $\varphi_{\mathrm{eff}}^{n,n+1}$
Single-Site Var.	Alg. 2.6	$\mathrm{O}(d\eta\chi^3) + \mathcal{C}_{\mathrm{init}}$	Contractions for $\varphi_{\mathrm{eff}}^{n}, L_n, R_n$
SVD compression	Alg. 2.4	$\mathrm{O}(d^2\eta^3\chi^3)$	QR decomposition
Zip-up	Alg. 2.7	$\mathrm{O}(d^2\eta\chi^3)$	dSVD

Table 2.1: Computational cost scaling for different MPO-MPS compression methods. We consider the scaling with the dimension d of the local Hilbert space, the bond dimensions χ of the state $|\psi\rangle$ to apply the operator O to, and the bond dimension η of that operator. We assume that we want to find an MPS approximation of $O|\psi\rangle$ with the same bond dimension χ, which is realistic e.g. during time evolution when the state $|\psi\rangle$ to be evolved already has the maximal allowed bond dimension. For the variational methods, $\mathcal{C}_{\mathrm{init}}$ denotes the cost of computing the initial guess, for which we have proposed either the other methods with costs listed here or methods with subdominant costs. We simplify the big O expressions, and decide which contribution is dominant under the mild assumptions $d \leq \eta$ and $d\eta \leq \chi$.

The variational compression methods can be generalized to approximate states as bounded bond dimension MPS in a more general setting, as long as the effective local states ϕ_{eff} of (2.47) or (2.50) can be computed efficiently. In particular, this applies to compressing states that are given as MPS already, to a smaller bond dimension, i.e. compressing them. We can achieve this by simply "leaving out" the MPO tensors from the diagrams above, or alternatively we can formally view the operator $O = \mathbb{1}$ with tensors $W^{[n]} \in \mathbb{C}^{1\times d\times d\times 1}$ given by $W^{[n]}_{\alpha,i,i',\beta} = \mathbb{1}_{i,i'} = \delta_{i,i'}$, such that its tensors can be contracted for free. This improves upon simply truncating the Schmidt values in a canonical form or iterating (2.24) in an isometric form.

2.4 Tensor networks in higher dimensions

A common way to approach the simulation of two-dimensional systems using tensor network methods is using MPS [95, 96]. Even though their intrinsic network connectivity and the variational power they are guaranteed by the entanglement paradigm are suggestive of a 1D geometry, the MPS algorithms, such as e.g. DMRG are exceptionally powerful and performant. As a result, MPS simulations have been established as state-of-the-art tools to simulate systems with entanglement beyond the 1D area law that the entanglement paradigm would otherwise restrict them to. This includes the study of 1D critical states [97], long-range interactions [90], two-dimensional systems [95, 96], as well as non-equilibrium dynamics [85, 93, 98–100]. It is particularly telling that rather than using the natively 2D ansaetze, such as PEPS, it is common to numerically approach 2D systems by winding an MPS around a thin cylinder, mapping the 2D model to a 1D geometry [96, 101–103].

This is done by choosing either a thin strip (with open boundary conditions in the thin direction) or a thin cylinder (with periodic boundary conditions) and treating the system as effectively 1D along the "long" direction, which we call the x direction or horizontal direction. There are two complementary perspectives on this approach. On the one hand, we can think of embedding the MPS in a snake geometry through the strip or cylinder or as a coil around the cylinder and understanding it as tensor network ansatz for a 2D quantum state, where virtual bonds only exist between some pairs of neighboring sites. Alternatively, we can think of mapping the 2D Hamiltonian to a 1D geometry, which comes at the price of increasing the range of horizontal couplings. The resulting algorithms typically have an unfavorable scaling with the vertical system size L_y, i.e. the width of the strip or circumference of the cylinder. This is because, for vertical bipartition, a 2D area law state has a bipartite entanglement that scales linearly with L_y and thus requires an MPS bond dimension *exponential* in L_y to approximate it up to some target precision. As a result, while the horizontal system size can easily be chosen large, or even infinite using iMPS, simulations are limited to thin systems, which can make extrapolation to the thermodynamic limit challenging. Nevertheless, these MPS methods are competitive with (and in many cases preferred over) the natively 2D methods because of superior performance and stability.

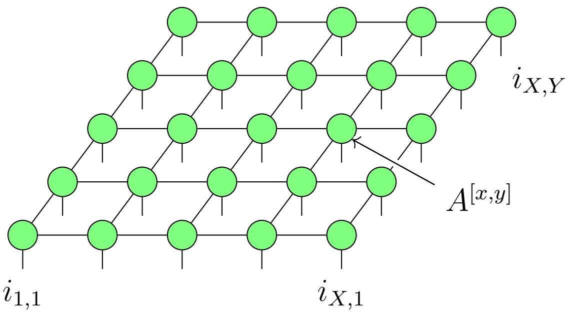
These 2D TNS, such as PEPS [25, 26] or MERA [22–24], on the other hand, are expected to well represent their class of target states – 2D area law states and 2D critical states respectively – at some finite bond dimension that does not scale with system size. While they have been successfully used to simulate 2D models, e.g. in Refs. [104–106], the advantage over MPS methods in practice is not as pronounced as the above would suggest. Algorithms to find good approximation within the variational manifold seem to be less powerful and less stable, and algorithmic development seems to have seen less progress compared to MPS methods. As demonstrated by the benchmark results in section 4.4, even for simple models such as the transverse field Ising model, established PEPS algorithms do not come close to finding the optimal PEPS at a given bond dimension. Even just evaluating observables in a given PEPS is a hard problem, both computationally and conceptually. It is an open question to what extent this discrepancy is due to a fundamental barrier induced by the network geometry and to what extent algorithmic improvements on the PEPS side can close the gap. We believe that the difficulty of fixing the gauge or establishing some sort of canonical form in TNS with loops is one of the greater hurdles, and approaches to partial gauge fixing have brought improvements [107, 108].

For the rest of this section, we discuss the PEPS ansatz for 2D systems, establish basic methods for evaluating observables, and briefly summarize established algorithms. This introduction also appears in the manuscript [2] by the author.

2.4.1 Projected Entangled Pair States (PEPS)

A projected entangled pair state (PEPS) is a TNS that is the natural generalization of MPS to higher dimensions. It results from the recipe of putting a tensor with a single physical leg on every site of the physical lattice and connecting neighboring tensors with a virtual leg. We assume a 2D square lattice, but higher-dimensional versions have been proposed [109, 110], and other lattices are accessible by either generalizing the ansatz [111] or mapping them to a square lattice. The ansatz

$$|\psi\rangle = \sum_{i_{1,1},\dots,i_{X,Y}}$$



$$|i_{1,1}, , \dots, i_{X,Y}\rangle \qquad (2.53)$$

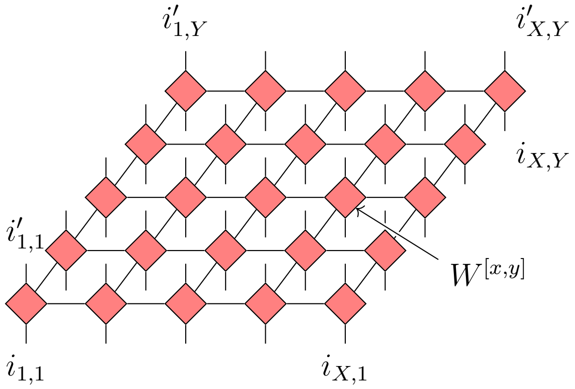
is given in terms of a five leg tensor $A^{[x,y]}$ for every lattice site (x,y). Throughout this section, we typeset graphical equations with a 5×5 system but understand this as a sketch with straightforward generalization to any system size $X \times Y$. Additionally, we do not label every object and refer to e.g. $A^{[x,y]}$ as a generic label instead of labeling all tensors $A^{[1,1]}, A^{[2,1]}, \dots, A^{[X,Y]}$. The tensors $A^{[x,y]}_{i_{x,y},u,l,d,r}$ each have a physical index $i_{x,y} \in \{1,\dots,d\}$ labeling an element $|i_{x,y}\rangle$ of an orthonormal basis for the local Hilbert space on site (x, y), and four virtual indices, and may be different on each site. At the boundary, the respective virtual indices that do not connect to a neighbor are assumed to be one-dimensional and can thus be removed.

An infinite version, called infinite projected entangled pair state (iPEPS) [112], can be obtained by repeating a rectangular unit cell of A tensors to tile the infinite plane. We focus on finite PEPS throughout this section.

Like for MPS, there is a gauge freedom for PEPS. While gauge fixing plays an important role in PEPS algorithms [107, 108], and analogs for a canonical form have been proposed [72, 113, 114], isometric properties analogous to (2.16) that establish an orthogonality center can in general not be achieved by using the gauge freedom. Enforcing 2D analogs of the isometric property and establishing an orthogonality center results in a strict subclass of PEPS, called isometric PEPS, or isometric TNS [70, 71].

In analogy to MPOs, defined in (2.30), we define projected entangled pair operators (PEPOs) as a tensor network with the same connectivity of virtual legs as a PEPS, but with two physical legs on each site, representing an operator on the many-body Hilbert space. Thus, a PEPO is an operator of the form

$$\sum_{\substack{i_{1,1}\dots i_{X,Y}\\ i'_{1,1}\dots i'_{X,Y}}}$$



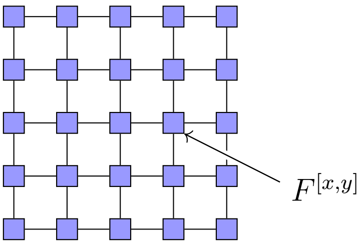
$$\left|i_{1,1}\dots i_{X,Y}\right\rangle\left\langle i'_{1,1}\dots i'_{X,Y}\right| . \tag{2.54}$$

Similar to the finite state machine construction for MPOs, schemes to write local Hamiltonians as PEPOs have been proposed [115, 116], and we provide an explicit construction for nearest neighbor models in section 4.4.1.

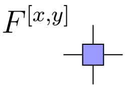
2.4.2 Contracting diagrams: boundary MPS method

Since we do not have access to an isometric property for PEPS, a simplification of the diagrams for norm or local expectation values similar to MPS, e.g. in (2.26), does not occur. As a result, e.g., the norm of a PEPS is given by

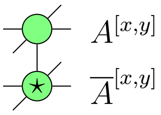
$$\langle\psi|\psi\rangle \quad =$$



$$;$$



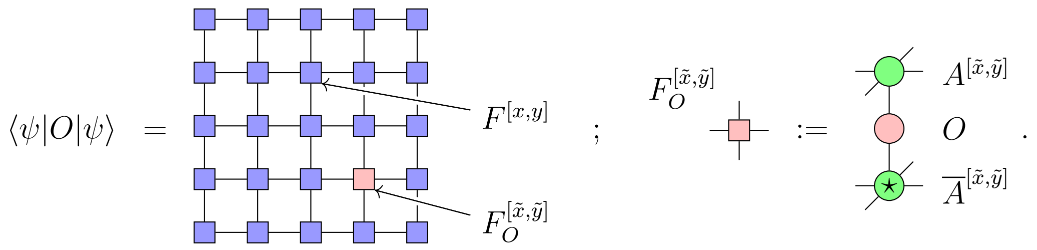
$$:=$$



$$, \tag{2.55}$$

where we defined the double layer tensor $F^{[x,y]}_{(uu')(ll')(dd')(rr')} := \sum_i A^{[x,y]}_{iuldr}\overline{A}^{[x,y]}_{iu'l'd'r'}$. The respective virtual legs pointing in the same lattice direction, e.g. u, u' pointing upward are combined to a single leg. Note that this composite object is mainly introduced for visualization purposes – to avoid three-dimensional diagrams – and should not be formed in practice. Any contraction involving $F^{[x,y]}$ should instead be accomplished by performing the contractions with $A^{[x,y]}$ and with $\overline{A}^{[x,y]}$ sequentially, which is cheaper.

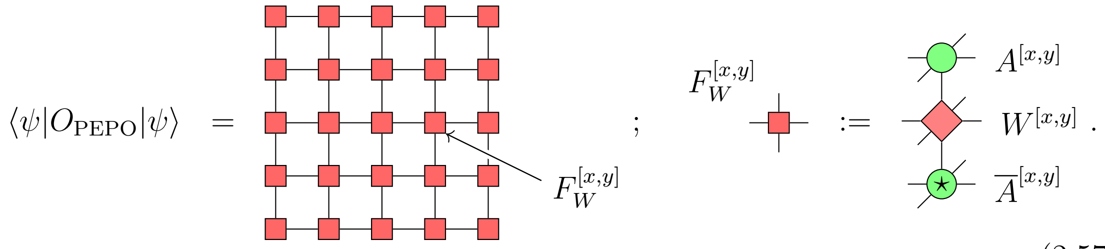
Expectation values, e.g. of a local operator O acting on a site $(\tilde{x}, \tilde{y})$, are achieved by modifying one of the respective double layer tensors



$\langle\psi|O|\psi\rangle$
$=$
$F^{[x,y]}$
$F_O^{[\tilde{x},\tilde{y}]}$
;
$F_O^{[\tilde{x},\tilde{y}]}$
:=
$A^{[\tilde{x},\tilde{y}]}$
O
$\overline{A}^{[\tilde{x},\tilde{y}]}$
.

(2.56)

The expectation values of a PEPO with tensors $\{W^{[x,y]}\}$ is given by a similar three-layer diagram

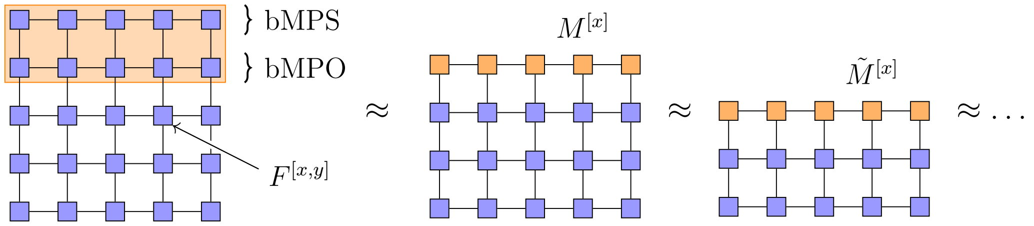


(2.57)

Overlaps $\langle\phi|\psi\rangle$ between different PEPS or matrix elements $\langle\phi|O|\psi\rangle$ of operators are structurally analogous and simply have different PEPS tensors in the ket (top) layer than in the bra (bottom) layer. Since all of these objects, norms, overlaps, expectation values, and matrix elements have the same structure of a square lattice of tensors and only differ in the details of their multi-layer structure, it is enough conceptually to focus on only the norm diagram.

Evaluating the norm, or an expectation value, is exponentially expensive in the linear system size [117]. If we consider, for example, an $L \times L$ system and a PEPS with bond dimension D and evaluate the norm via pairwise tensor contraction, we will eventually encounter an intermediate tensor with an extensive number $\sim L$ of open D-dimensional legs. Its number of entries is exponential in L, e.g. typically D^{2L+2}, and thus the memory cost to store it and the floating point operation (FLOP) count to perform the next contraction are exponential as well. This is unfeasible, even for moderate system sizes, and the goal of evaluating these quantities exactly must be relaxed. Some approaches resort to Monte Carlo style sampling to obtain, e.g. expectation values [118, 119]. Here, we focus on introducing approximations on the tensor network level, and in particular, the boundary matrix product state (bMPS) contraction method [26, 107, 120]. The intuition behind bMPS contraction is to view the rows[4] of the diagram as either a state (for the top row) in a virtual Hilbert space of the PEPS bonds or an operator on it (for the bulk rows). Upon collapsing the multi-layer structure, the top row is a tensor network with the structure of an MPS, and it is commonly called a boundary matrix product state (bMPS). In analogy, we refer to the bulk rows as bulk matrix product operators (bMPOs), which are MPOs with additional multi-layer structure.

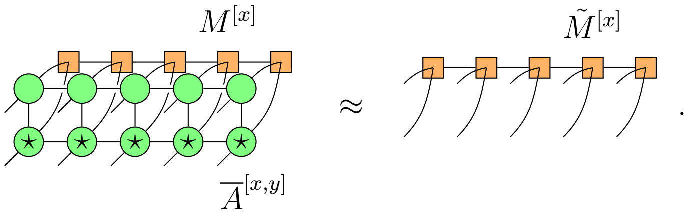
Evaluating a PEPS diagram, such as e.g. the norm $\langle\psi|\psi\rangle$ or an expectation value $\langle\psi|O|\psi\rangle$ thus amounts to applying a sequence of bMPOs to a bMPS. While rigorous justifications require further assumptions [120, 121], it is observed in practice that for relevant PEPS arising e.g. as candidates during ground state search of some 2D local, gapped Hamiltonian, the intermediate states on the virtual Hilbert space remain lowly entangled and can therefore be well-approximated by a bMPS of bounded bond dimension χ. Observables, and in particular the variational energy as a measure of the quality of the variational ground state approximation, should, in practice, be verified by a careful scaling analysis with χ. Sequentially applying the bMPOs, introducing approximations where necessary to keep the bMPS bond dimension bounded gives us a scheme



(2.58)

to evaluate PEPS diagrams. Note that the orange bMPS tensors $\{M^{[x]}\}$ or $\{\tilde{M}^{[x]}\}$ do *not* have a double layer structure. Therefore, the bMPS in the norm diagram, preserve the property of positivity as a map from the bottom layer legs associated with the bra PEPS to the top layer (ket) legs only approximately. While preserving positivity exactly would be desirable, enforcing it leads to inferior algorithms, see e.g. discussion in [107, 120].

Let us now focus on the problem of (approximately) applying a bMPO to a bMPS that is



(2.59)

We can achieve this using the methods developed in the context of MPO evolution, discussed in section 2.3.5. The multi-layer structure of the bMPO makes no difference conceptually, and in practice only needs to be considered in the contraction orders. Whenever an object needs to be contracted with a bMPO tensor $F^{[x,y]}$, it

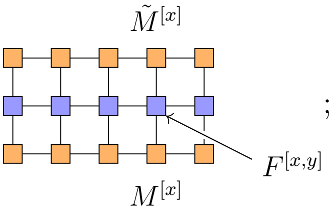
[4] This is a conventional choice. The contraction can also be analogously carried out column-wise. Which way is better may depend on the details of the system size and bond dimensions of the state or the operators, which may have anisotropic bond dimensions.

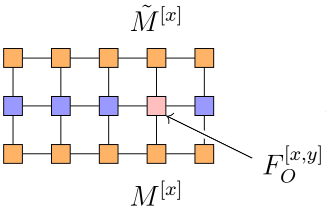
should be contracted with its factors layer by layer. We give the computational cost scaling in table 2.2. These are essentially the costs of MPO evolution summarized in table 2.1 where "physical" dimension and bMPO dimension coincide as D^2 for the norm and ηD^2 for the expectation value, but takes into account that contractions are cheaper due to the multi-layer structure.

Method	norm	expectation value	dominant step(s)
Two-Site Var.	$\mathrm{O}(D^6\chi^3) + \mathcal{C}_{\mathrm{init}}$	$\mathrm{O}(\eta^3 D^6\chi^3) + \mathcal{C}_{\mathrm{init}}$	Contractions & dSVD
Single-Site Var.	$\mathrm{O}(D^4\chi^3) + \mathcal{C}_{\mathrm{init}}$	$\mathrm{O}(\eta^2 D^4\chi^3) + \mathcal{C}_{\mathrm{init}}$	Contractions & QR
SVD compression	$\mathrm{O}(D^{10}\chi^3)$	$\mathrm{O}(\eta^5 D^{10}\chi^3)$	QR
Zip-up	$\mathrm{O}(D^6\chi^3)$	$\mathrm{O}(\eta^3 D^6\chi^3)$	dSVD

Table 2.2: Computational cost scaling for PEPS contraction using the bMPS method. The methods are the same MPO evolution methods as in table 2.1. We give the cost for (a) evaluating the norm of a PEPS with bond dimension D and (b) the expectation value of a PEPO with bond dimension η in such a PEPS, both at a (maximum) bMPS bond dimension χ. We have simplified the scalings under the assumptions $dD^2 \leq \chi^2$ and $d\eta^2 \leq D^2$. Note that using the multi-layer structure of the bMPO reduces the cost of contraction steps, e.g. down to the same scaling as the decompositions for the variational methods. Here, $\mathcal{C}_{\mathrm{init}}$ denotes the cost of computing the initial guess for the variational methods, for which we propose either the zip-up method, a random bMPS, or the bMPS tensors from a related PEPS contraction in a previous iteration of an outer loop of a PEPS algorithm.

Returning to the task of contracting an entire diagram (2.58), we can evaluate the norm of a PEPS, or an expectation value by absorbing rows into a top or bottom bMPS, until all but one row of the diagram are absorbed. Then, we can evaluate them as

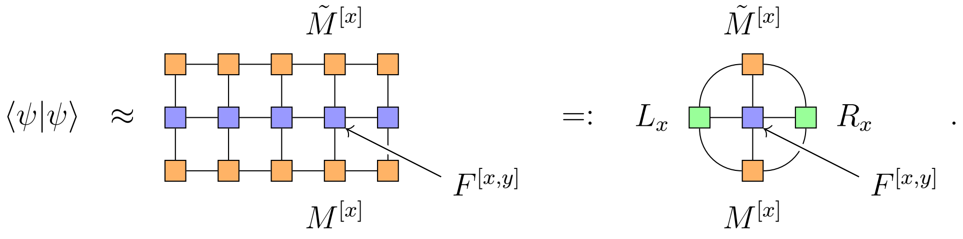
$\langle\psi|\psi\rangle \approx$ 

; $\langle\psi|O|\psi\rangle \approx$ 

, (2.60)

via pairwise contraction. The cost of this contraction is in $\mathrm{O}(D^4\chi^3)$ for the norm diagram of a local operator and in $\mathrm{O}(\eta^2 D^4\chi^3)$ for a three-layer diagram with a PEPO. This is the same scaling as for the cheapest method to perform the bMPS-bMPO absorption, single-site variational sweeping, and thus never dominant. Note that for a local operator O it is convenient to sandwich the row on which the operator acts, since then the bMPS are the same as for the norm diagram and may be re-used.

From the bMPS contraction, we get access to the *effective environment* of a given site (x, y), which consists of the four tensors surrounding $F^{[x,y]}$ on the RHS of



(2.61)

It is a central object in variational updating algorithms, and may also be evaluated using different approximate contraction schemes, such as e.g. corner transfer renormalization group (CTMRG) [122].

2.4.3 PEPS ground state search algorithms

Let us now review algorithms to find a good approximation to the ground state of a Hamiltonian within the manifold of PEPS of a given bond dimension D. We focus on methods that apply to finite systems.

In the original PEPS work in Ref. [26], the authors propose both a variational energy minimization, as well as an imaginary time evolution approach. The variational approach is conceptually related to the single site DMRG algorithm, iterating single-site updates which keep all other tensors constant, but crucially without an isometric form. As a consequence, the local problem, analogous to (2.39) comes with a non-trivial effective norm environment N_{eff} arising from (2.61) and finding its optimal solution requires solving a generalized eigenvalue problem $H_{\text{eff}} \cdot \theta = \varepsilon N_{\text{eff}} \cdot \theta$, which is often ill-conditioned. We expect these single-site updates without mixing approaches to suffer from similar convergence issues as single-site DMRG. The imaginary time evolution approach involves applying a global approximation $\mathrm{e}^{-H\delta\tau} \approx \mathbb{1} - H\delta\tau$ of the time evolution operator, and subsequently truncating the PEPS bonds variationally, similar to variational MPO evolution, see section 2.3.5. Again, since there is no canonical form established, the local problem involves a non-trivial effective norm matrix and is often ill-conditioned.

The FU algorithm [112] implements imaginary time by applying the two-body gates arising from a Trotter decomposition sequentially, i.e. one at a time. Many adaptations of the FU, including reduced local updates [107, 123], as well as gauge fixing [108, 120] have been employed to improve stability and performance. The simple update (SU) algorithm [124], on the other hand can be viewed as a version of the FU that makes strong simplifications of the effective norm environment. As a result, updates are in general further from optimality, since they are derived from a norm that is a worse approximation of the true many body norm, but are significantly cheaper, and thus allow larger bond dimensions. Thus, the SU allows ground state search in a larger variational manifold, that generically contains a better approximation to the true ground state, but the search algorithm is inferior and generically finds a worse candidate then a FU search in the same manifold would. It is an open question, under what circumstances the trade-off introduced by relaxing to a SU is worth it. Note that (generically), the FU does not find the optimal ground state approximation in its search manifold either, as we demonstrate in section 4.4.

Gradient-based approaches have been employed for both finite [115, 125–128] or infinite systems [37–39, 129], and minimize the variational energy, by globally updating all tensors at once. For infinite systems, it is common to evaluate gradients using automatic differentiation (AD), which seems to work well and give stable optimization trajectories. For finite systems, a similar approach seems to interact poorly with the approximations needed to evaluate the variational energy, e.g. using the bMPS method. We propose an approach to stable gradient-based optimization of finite PEPS in chapter 4.

2.4.4 PEPS time evolution algorithms

Regarding simulation of time evolution, *infinite* PEPS simulations typically employ sequential gate application [130, 131]. The derivation of the update conceptually breaks the translational invariance of the ansatz, by assuming that the infinite environment surrounding a unit cell remains fixed, while deriving a local update for a few, typically two, tensors in the unit cell. Translational invariance is then restored by performing the so derived update in all unit cells. This is an approximation to a truly translationally invariant application of the gate to all unit cells simultaneously. In addition to the FU and SU environment schemes, a significant performance improvement was gained by the invention of the fast full update (FFU) [108], which exploits that the state does not change much under small time steps and thus new environments can be obtained from the old environments in only a single CTMRG [132] renormalization step.

While the original PEPS work [26] mentions applying the evolution approach also to real time dynamics, we are not aware of any published works simulating real-time dynamics of *finite* PEPS, and we have not been able to obtain sensible results from variational schemes with local updates.

Simulating local quench dynamics, e.g. to extract response functions, however, requires either a finite system, or a translationally invariant state with a large unit cell. In either case, the system is expected to well-approximate the thermodynamic limit for a limited time, until correlations from the initial quench have spread to the boundary of the finite system or the unit cell. The approach using infinite systems with a large unit cell (15×15) is done in Ref. [133], using sequential time evolution with local updates derived from FFU environments. Let us mention for completeness the approach of Ref. [134] to simulate local excitations using a translationally invariant superposition, employing SU simulations with a small (2×2) unit cell.

We propose a gradient-based time evolution algorithm for finite PEPS in chapter 4.

2.5 Symmetries

If the Hamiltonian has a symmetry, that is, if it commutes with a unitary U, it has a block-diagonal structure in the eigenbasis of the symmetry. This block-diagonal structure can be exploited for computational performance in both memory and FLOPs, as fewer numbers need to be stored for such symmetric matrices, and operations on them, such as contractions and decompositions can be done blockwise and require fewer FLOPs.

In this section, we focus on a "global symmetry", that is a unitary representation $U^{(n)}$ of a symmetry group G on every local Hilbert space at site n, such that the Hamiltonian is invariant under the global action $[H, \otimes_{n=1}^{N} U^{(n)}(g)] = 0$ for every group element $g \in G$. The assumption that the symmetry factorizes into on-site unitaries, i.e. that it has a spatial structure, allows it to cooperate with the tensor networks which have an inherent spatial structure as well. We introduce a more general framework that goes beyond symmetries induced by group representations in chapter 5, but discuss the group case here as an instructive special case and restrict to abelian groups for the concrete consequences on the tensor level.

We give a detailed review of the basics of representation theory and state central results such as Schur's lemma in section A.1. The main points are that first, any unitary representation decomposes into the direct sum of irreducible representations – irreps for short. For each group, G, the irreps can be classified and we write $U_{\mathbf{a}}$ for the representative of an equivalence class of irreps, where the labels are e.g. $\mathbf{a} \in \mathbb{Z}$ for $G = \mathrm{U}(1)$ or $\mathbf{a} \in \mathbb{Z}_N$ for $G = \mathbb{Z}_N$. Secondly, Schur's lemma part 1 states that equivariant maps, that is, linear maps $f : V \to W$ that are compatible with representations U_V (U_W) on V (W) in the sense that $f \circ U_V(g) = U_W(g) \circ f$ for all $g \in G$ must vanish if $U_V \ncong U_W$ are inequivalent and irreducible. Lastly, the tensor product $U_{\mathbf{a}} \otimes U_{\mathbf{b}}$ of irreps is itself irreducible and thus equivalent to an irrep $U_{\mathbf{a}+\mathbf{b}}$, which defines the addition rule for irrep labels.

As an example, consider the spin-$\frac{1}{2}$ Heisenberg chain in a field, with Hamiltonian

$$H = \sum_{n=1}^{N-1} \boldsymbol{S}_n \cdot \boldsymbol{S}_{n+1} + h^z \sum_{n=1}^{N} S_n^z. \tag{2.62}$$

It has a U(1) symmetry, which conserves the total magnetization $Q = \sum_n S_n^z$. For completeness, let us state the concrete representation

$$U^{(n)}(\mathrm{e}^{\mathrm{i}\phi}) = \mathrm{e}^{\mathrm{i}\phi 2 S_n^z/\hbar} = \mathrm{e}^{\mathrm{i}\phi} \left|\uparrow\right\rangle\!\left\langle\uparrow\right| + \mathrm{e}^{-\mathrm{i}\phi} \left|\downarrow\right\rangle\!\left\langle\downarrow\right|, \tag{2.63}$$

where $\mathrm{e}^{\mathrm{i}\phi} \in \mathrm{U}(1)$ is a general group element. We find that the representation decomposes as $U^{(n)} = U_{\mathbf{1}} \oplus U_{-\mathbf{1}}$, where $U_{\mathbf{n}}$ are the representative irreps of U(1) labelled by integers $\mathbf{n} \in \mathbb{Z}$, see section A.3. Note that this singles out the z basis as an advantageous computational basis since the representation decomposes directly into irreps in this basis, which it would, e.g. not do in the x basis. This is a common pattern; a symmetry group implies a canonical choice for the computational basis. In practice, we do not need to work with the representation explicitly and only care about the irrep label for each basis element, in this case $\mathbf{1}$ for $|\uparrow\rangle$ and $-\mathbf{1}$ for $|\downarrow\rangle$.

2.5.1 Symmetric States from Symmetric Tensor Networks

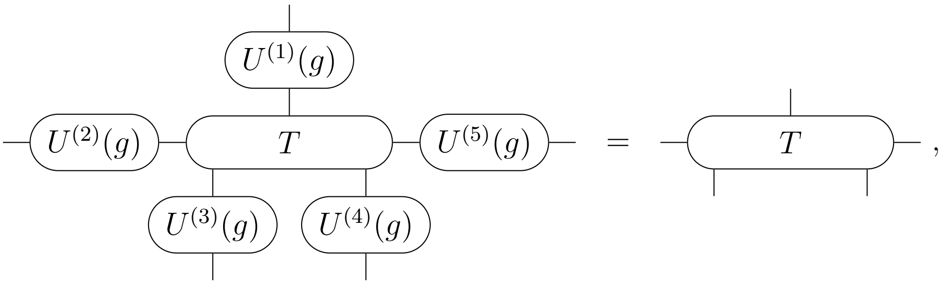
Let us first focus on symmetric states that is, many-body states $|\psi\rangle$ that are invariant

$$\otimes_{n=1}^{N} U^{(n)}(g)\,|\psi\rangle = |\psi\rangle \tag{2.64}$$

under the group action. We extend the resulting framework to covariant states (that transform non-trivially under the symmetry) afterward. Now, what are the implications for tensor network representations of $|\psi\rangle$?

First, let us consider one particular way to force tensor networks to yield symmetric states, namely by building them from symmetric tensors. First, we require a representation of the symmetry group on each virtual Hilbert space of the tensor network. Up to equivalence, which can be safely gauged away, the only relevant property of this representation is its irrep content, that is, which irreps appear and how often. We can view this as choosing a separate bond dimension of the virtual Hilbert space in every symmetry sector, a choice that influences the variational power of the ansatz. In algorithms where bond dimensions can dynamically grow, such as in most MPS algorithms, this never needs to be chosen manually since the Schmidt spectrum assigns priorities to the respective basis states and allows a clear choice, which irreps to keep.

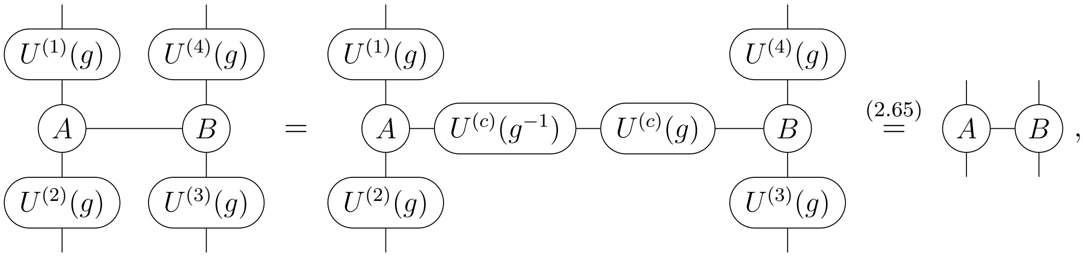
Given such representations, a tensor T is *symmetric* if it is invariant under the simultaneous group action on all of its legs, that is, for all $g \in G$



(2.65)

where we have drawn a five leg example tensor for concreteness.

Now if two symmetric tensors are contracted, the contracted legs need to be compatible, in the sense that one is the dual of the other, i.e. one is "bra-like", while the other is "ket-like". As a result, the representation $|\psi\rangle \mapsto U(g)|\psi\rangle$ on one leg is cancelled by the contragradient representation $\langle\phi| \mapsto \langle\phi|U^{\dagger}(g) = \langle\phi|U(g^{-1})$ on the dual leg



(2.66)

such that the contraction results in a composite tensor, which is itself symmetric. Thus, a tensor network consisting of symmetric tensors gives a symmetric state.

For loop-free tensor networks, such as MPS, we can also argue the other way, namely that if a symmetric state can be written as a tensor network at all, then it can be written as a tensor network *of symmetric tensors* at the same bond dimensions. For finite MPS, e.g., we can formally see this by performing a sequence of SVDs of the full wave function. At each SVD, the number of non-zero singular values and, thus, the resulting rank is the same as in the network of non-symmetric tensors. For infinite MPS, see e.g. the discussion in [40, Sec. III.A.1]. As a result, we do not lose any variational power by restricting to networks of symmetric tensors.

2.5.2 Symmetric Tensors

In this section, we establish a parameterization for symmetric tensors. This allows us to store only the free parameters when storing tensors and operate only on those parameters when manipulating tensors instead of all entries of general, non-symmetric tensors. We assume that the symmetry group is abelian at this point; see chapter 5 for the non-abelian case. There are two routes to obtain these results commonly used in the literature, either using the Wigner-Eckart theorem or Schur's Lemma. We follow the latter route, but the consequences are the same.

Since Schur's lemma is a statement about maps, we recast tensors as linear maps. In particular a tensor $T \in V_1 \otimes \dots \otimes V_N$ is equivalent to a linear map

$$t : \mathbb{C} \to V_1 \otimes \dots \otimes V_N, \alpha \mapsto \alpha T \tag{2.67}$$

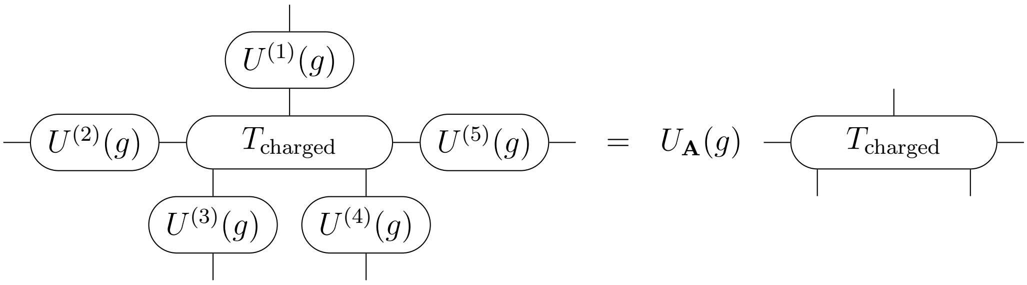
and we can recover $T = t(1)$. Now, T is a symmetric tensor w.r.t. representations $U^{(n)}$ on each of the spaces V_n if and only if t is an equivariant map between the trivial representation $U_\mathbf{0}$ on $\mathbb{C}$ and $\bigotimes_n U^{(n)}$ on $\bigotimes_n V_n$. Let us assume that on every leg n of the tensor we have chosen the computational basis such that the group representation is a direct sum of irreps $U^{(n)} = \bigoplus_{i=1}^{\dim V_n} U_{\mathbf{a}_i^{(n)}}$ which defines $\mathbf{a}_i^{(n)}$ as the irrep label of the i-th component in the decomposition of $U^{(n)}$. A general entry $T_{i_1,\dots,i_N}$ is now an equivariant map between $U_\mathbf{0}$ and $\bigotimes_{n=1}^N U_{\mathbf{a}_{i_n}^{(n)}} \cong U_{\mathbf{a}_{i_1}^{(1)}+\dots+\mathbf{a}_{i_N}^{(N)}}$. By part 1 of Schur's lemma – see section A.1 – it can only be non-zero if these representations are equivalent. This gives rise to the charge rule for symmetric tensors

$$\mathbf{a}_{i_1}^{(1)} + \mathbf{a}_{i_2}^{(2)} + \dots + \mathbf{a}_{i_N}^{(N)} \neq \mathbf{0} \quad \Rightarrow \quad T_{i_1,i_2,\dots,i_N} = 0. \tag{2.68}$$

As a consequence, symmetric tensors have a sparsity structure, where a significant fraction of entries are forced to vanish by charge conservation. Now, if we choose the order of the computational basis such that those indices i_n that have the same irrep label $\mathbf{a}_{i_n}^{(n)}$ appear consecutively, this results in a block-sparse structure of the tensors. Thus, it is enough to store only those non-zero blocks in memory. Additionally, operations on tensors reduce to operations on the smaller blocks, with some additional book-keeping to identify the indices corresponding to the resulting blocks. This applies to combining and splitting legs, contracting tensors, decomposing them in e.g. SVDs, and more. We do not go into detail regarding implementation at this point and refer the interested reader to the literature [35, 51, 135] since we develop a more general framework that covers the abelian groups as a special case and describe operations on tensors in detail, in chapter 5.

For a bra space, it is convenient to keep track of the irreps of its dual ket space instead, which differ from its own irreps by a minus sign. This is because the irreps are sorted for performance optimization, and this convention results in a compatible order in the dual space, which facilitates contraction. As such, the irrep for the i-th index on a bra space is $-\mathbf{a}_i$, where $\mathbf{a}_i$ is the irrep of the i-th index on the corresponding ket space. This introduces explicit signs $\zeta^{(n)} = \pm 1$ for each leg to the charge rule, indicating if the respective leg is a ket space ($+1$) or bra space (-1).

Additionally, it is convenient to introduce a "total charge", which is an irrep label $\mathbf{A}$ and replace the trivial irrep $U_\mathbf{0}$ on the domain $\mathbb{C}$ by $U_\mathbf{A}$. Now if the map t is equivariant between $U_\mathbf{A}$ and $\bigotimes_{n=1}^{N} U_{\mathbf{a}^{(n)}_{i_n}}$, this means that the tensor transforms under $U_\mathbf{A}$ if the symmetry is applied, meaning



(2.69)

for all $g \in G$. Here, $U_\mathbf{A}(g)$ is just a complex phase, as it is a unitary representation on $\mathbb{C}$. We call such tensors *charged* (as opposed to symmetric).

The resulting charge rule, with explicit signs and a total charge, is then

$$\zeta^{(1)}\mathbf{a}^{(1)}_{i_1} + \zeta^{(2)}\mathbf{a}^{(2)}_{i_2} + \cdots + \zeta^{(N)}\mathbf{a}^{(N)}_{i_N} \neq \mathbf{A} \quad \Rightarrow \quad T_{i_1,i_2,\ldots,i_N} = 0. \tag{2.70}$$

It is often directly formulated in terms of charges instead of irrep labels. Charge values are eigenvalues $q(\mathbf{a}_i)$ of a conserved charge operator of the form $Q_n = \sum_i q(\mathbf{a}_i)|i\rangle\langle i|$, i.e. an operator diagonal in the basis induced by the irrep decomposition, whose diagonal entries depend on the irrep label such that irreps have a *unique* charge value. For a concrete example, consider a spin system with the U(1) symmetry that conserves $Q_n = S^z_n$ in the sense that $[H, \sum_n Q_n] = 0$. We find $Q_n = S^z_n = \sum_i \frac{\hbar}{2}\mathbf{a}^{(n)}_i|i\rangle\langle i|$. Recall that for U(1), we choose integer irrep labels $\mathbf{a} \in \mathbb{Z}$. In the case of U(1) specifically, the conserved charge is up to a prefactor the representation of the Lie algebra generator, such that the representation of the group is given by $U^{(n)}(\mathrm{e}^{\mathrm{i}\phi}) = \mathrm{e}^{\mathrm{i}\phi 2S^z_n/\hbar}$. From this perspective, a symmetric state

$$\bigotimes_n U^{(n)}\,|\psi\rangle = |\psi\rangle \quad \forall g \in G \quad \Rightarrow \quad Q\,|\psi\rangle = q(\mathbf{0})\,|\psi\rangle \tag{2.71}$$

is an eigenstate of the conserved charge with eigenvalue $Q = q(\mathbf{0})$, which is typically[5]

[5] The zero-point of Q is arbitrary since constants can be added to get a conserved charge that is just as valid. It is common to set the zero point to correspond to the symmetric state, e.g. zero S^z magnetization or zero particle number w.r.t. some reference filling fraction.

but not necessarily $Q = 0$. A charged state, on the other hand

$$\bigotimes_n U^{(n)} |\psi\rangle = U_{\mathbf{A}}(g) |\psi\rangle \quad \forall g \in G \quad \Rightarrow \quad Q |\psi\rangle = q(\mathbf{A}) |\psi\rangle \tag{2.72}$$

is an eigenstate with a different eigenvalue. Thus, it can e.g. be used to parametrize states with finite magnetization, or if the U(1) symmetry conserves particle number, with finite filling.

Of particular interest in this thesis is the special case where we only have two legs (or legs have been grouped into two groups), i.e. symmetric matrices. This is the form that allows decomposition, e.g. SVDs of symmetric tensors, by first grouping legs to a matrix, decomposing it, and subsequently ungrouping to recover the original leg structure, see e.g. (2.9). We focus on only the central step, decomposing symmetric matrices. We assume that the computational basis is sorted by charge sectors/irreps. As a result, those irrep labels $\mathbf{a}_q$ with $q = 1, \ldots, Q$ that appear on *both* legs of a matrix θ each correspond to a block θ_q, and all other entries – not in one of these blocks – vanish by the charge rule. We obtain a block-sparse structure, such as e.g

$$\theta \;=\; \begin{pmatrix} \boxed{\theta_1} & & & 0 \\ & \boxed{\theta_2} & & \\ & & \boxed{\theta_3} & \\ 0 & & & \boxed{\theta_4} \end{pmatrix} . \tag{2.73}$$

Note that each row (column) contains at most one block, and there may be rows (columns) that do not have a block, if the irrep associated with the index of that row (column) has no matching irrep in the other leg. We refer to the sizes $m_q \times n_q$ of the blocks θ_q as the block-sizes of θ.

We can then achieve standard decompositions of these matrices by acting blockwise, such as e.g. algorithm (2.8) for an SVD.

Algorithm 2.8 Truncated SVD for block-diagonal matrices

Given an $m \times n$ block-sparse matrix θ with block sizes $\{m_q \times n_q\}$ and a target rank $k \leq \min(m, n)$, compute the truncated SVD $\theta \approx USV^\dagger$ consisting of block-diagonal left isometries U, V, and S the diagonal matrix containing the k largest singular values of θ.

1. For every block q, compute the SVD $\theta_q =: U_q S_q V_q^\dagger$ with rank $k_q = \min(m_q, n_q)$.
2. Identify a threshold value $\lambda \geq 0$, such that $\sum_q |\{i = 1, \ldots, k_q | (S_q)_{ii} > \lambda\}| = k$, i.e. such that exactly k singular values are larger.
3. For every block q, truncate its SVD to $\theta_q \approx \tilde{U}_q \tilde{S}_q \tilde{V}_q^\dagger$, by keeping only singular values from S_q which are above λ and corresponding columns of U_q, V_q.
4. Form U, S, V from the blocks $\{\tilde{U}_q\}, \{\tilde{S}_q\}, \{\tilde{V}_q\}$

This can clearly be modified to incorporate additional criteria for which singular values to keep. The important part is that the selection of singular values to keep should be coordinated between the blocks.

2.6 The Tensor Network Python (TeNPy) library

The Tensor Network Python (TeNPy) library [4] is a python package for tensor network simulations. The library is based on previous (non-public) codes used in references [136, 137], developed by the authors of these works. It was rewritten in its current structure by Johannes Hauschild during his PhD and released as an open source package [135]. The package is currently maintained by Johannes Hauschild and the author of this thesis, and has recently seen a version 1.0 release [3]. Current active development aims to improve the low-level part of the package handling linear algebra of tensors, to support nonabelian symmetries, generalized symmetries, and fermionic or anyonic degrees of freedom – see chapter 5 – as well as hardware acceleration on GPUs, to incorporate e.g. the concepts discussed in chapter 3. A prototype of the new implementation was developed by the author of this thesis, and is currently being optimized for performance and incorporated into the rest of the library.

The package offers high-level functionality that abstracts entire simulations of, e.g. a response function from time evolution or a phase diagram sweep that performs ground state search at many parameter values and extracts order parameters. These are implemented in terms of "mid-level" algorithms, such as e.g. the DMRG, TEBD and MPO evolution algorithms discussed in sections 2.3.3-2.3.5 respectively, as well as TDVP and VUMPS. The focus is currently mostly on MPS simulations in one and two spatial dimensions. Many features exist to facilitate the specification of the physical models, in terms of a 1D, quasi-1D, or 2D lattice geometry. As a result, Hamiltonians can be specified in the natural language of local operators and couplings on the lattice, and the construction of e.g. MPO representations of the Hamiltonian is fully automated. This allows non-experts to set up, e.g. a DMRG simulation for a particular physical model without knowing any details of the tensor network representation of its Hamiltonian, or run TEBD without interacting with the Trotterization and the order in which gates are applied. The low-level functionality offers linear algebra routines on symmetric tensors, supporting abelian symmetry groups, as discussed in section 2.5.

The code is open source and maintained publicly on GitHub [4]. Refer to the online documentation[6] to get started.

[6] `https://tenpy.readthedocs.io/en/latest/`

Chapter 3

Fast time evolution of matrix product states

This chapter, and in particular sections 3.2 and 3.5, are based on a previous publication of the author [1].

A common pattern in TNS algorithms are truncation steps, where an update that is derived in an enlarged search space needs to be approximated within the variational manifold. For example, the two-site updates of MPS during TEBD or two-site DMRG, as described in sections 2.3.3 and 2.3.4 respectively, lie outside the manifold of fixed bond-dimension MPS and thus need to be truncated. In most cases, this requires an approximate matrix factorization $\theta \approx E^{[n]} \cdot F^{[n+1]}$ with a bounded rank of at most k and is typically implemented with a truncated SVD. Additional properties, e.g. isometric properties of some factors can be desirable in many settings.

In this chapter, we discuss several alternative factorizations that (a) have lower cost scaling, (b) allow for hardware acceleration, or (c) stabilize automatic differentiation. In order to discuss the advantages and disadvantages of these matrix factorization routines, we can mostly take the point of view of general-purpose numerical linear algebra. There are two important aspects to consider, however, if we come from the context of tensor networks, and in particular MPS. Firstly, we need to be able to both deal with and exploit the block-sparse structure arising from symmetries. Secondly, from the TNS algorithm, we may assume that we have access to a related matrix $\hat{\theta}$ which is close to θ and has an exact factorization $\hat{\theta} = \hat{E}\hat{F}$ with rank $\hat{k} \leq k$. This is natural in tensor networks since the purpose of the truncation is to restore the factorized TNS form that was present before the update. Additionally, the pre-update $\hat{\theta}$ is naturally close to the update θ, e.g. in TEBD where it differs by time evolution by a small time step $\hat{\theta} = \theta + \mathrm{O}(\delta t)$, or in variational algorithms such as DMRG, if they are close to convergence.

First, in section 3.1, we summarize properties of standard matrix factorizations and establish categories for truncated factorizations with the minimal properties to be useful in tensor network simulations. We introduce the QR-based factorization and its algorithmic variations in section 3.2 and discuss its relation to randomized linear algebra in section 3.3. In section 3.4, we propose synthesized routines, incorporating elements from randomized linear algebra in the QR-based scheme, perform a benchmark in section 3.5 and conclude in section 3.6.

3.1 Matrix factorizations

This chapter deals extensively with matrix factorizations. In this section, we therefore first establish different (categories of) factorizations and summarize their properties. We will place a particular focus on truncated factorizations, that is, low-rank approximations of the input matrix with particular properties analogous to the corresponding exact factorization. We give automatic differentiation (AD) formulae, in particular in light of truncation and for the deformed version of the SVD in section 4.2.

3.1.1 Singular Value Decomposition (SVD)

The singular value decomposition (SVD) is ubiquitous in tensor networks. It is the most direct and provably optimal way to find a truncated decomposition of a general input matrix. An SVD of an $m \times n$ matrix θ is a factorization

$$
\begin{aligned}
&\theta = USV^\dagger \qquad \text{(SVD)} \\
&\qquad U \in \mathbb{C}^{m\times k} \quad \text{left isometry: } U^\dagger U = \mathbb{1} \\
&\qquad S \in \mathbb{R}^{k\times k} \quad \text{real, non-negative, diagonal, non-increasing:} \\
&\qquad \qquad \qquad S_{i,i} \geq S_{i+1,i+1} \geq 0 \text{ and } S_{i,j} = 0 \text{ if } i \neq j \\
&\qquad V \in \mathbb{C}^{n\times k} \quad \text{left isometry: } V^\dagger V = \mathbb{1}\ ,
\end{aligned}
\tag{3.1}
$$

where $k = \min(m, n)$, as this is the “reduced” or “economic” version. Computing it has a runtime complexity in $\mathrm{O}(mnk)$. It seems that currently available implementations for SVD on GPUs are inefficient, preventing any benefit from hardware acceleration. In fact, in the benchmark in section 3.5, we find that while SVD-free algorithms have a speedup of almost two orders of magnitude on GPU, comparing a specific pair of GPU and CPU models, a similar algorithm that *is* using SVDs heavily, is actually slower on the GPU.

An SVD can be truncated to a target rank $\chi \leq k$ by taking slices such that only the first χ singular values are kept, i.e. defining $\tilde{U} = U_{[:,:\chi]}$, $\tilde{S} = S_{[:k,:k]}$ and $\tilde{V} = V_{[:,:\chi]}$. As a result, we have an approximate factorization $\theta \approx \tilde{U}\tilde{S}\tilde{V}^\dagger$, such that the truncation error

$$
\varepsilon_{\text{SVD}} = \left\|\theta - \tilde{U}\tilde{S}\tilde{V}^\dagger\right\|_{\text{F}} = \sqrt{\sum_{i=\chi+1}^{k} S_{i,i}^2} = \left\|S_{[\chi:,\chi:]}\right\|_{\text{F}}
\tag{3.2}
$$

is provably the lowest possible error for a rank χ factorization. As such, the SVD is the gold standard for low-rank factorizations.

3.1.2 QR Decomposition

A QR decomposition of a tall rectangular $m \times n$ matrix θ where $m \geq n$ is a factorization

$$\begin{aligned} &\theta = QR \qquad (\text{QR}) \\ &\quad Q \in \mathbb{C}^{m\times n} \quad \text{left isometry: } Q^\dagger Q = \mathbb{1}_n \\ &\quad R \in \mathbb{C}^{n\times n} \quad \text{upper triangular: } R_{i,j} = 0 \text{ if } i > j. \end{aligned} \tag{3.3}$$

Its computational cost has the same scaling $\mathrm{O}(mn^2)$ as an SVD of the same input matrix. The prefactor suppressed by the big O notation is, however, commonly much smaller. There are efficient GPU implementations. There is no direct way to truncate a standard QR decomposition, at least not with a controlled truncation error; see the modified variant described in the next section. While the upper triangular property of the R factor is needed in many numerical linear algebra applications, such as solving linear systems or least squares problems, in tensor networks, we typically only care about the isometric property of the Q factor and the reduced dimension of the new index.

The related LQ decomposition of wide rectangular $m \times n$ matrix θ where $m \leq n$ has the properties

$$\begin{aligned} &\theta = LQ \qquad (\text{LQ}) \\ &\quad L \in \mathbb{C}^{m\times m} \quad \text{lower triangular: } L_{i,j} = 0 \text{ if } i < j. \\ &\quad Q \in \mathbb{C}^{m\times n} \quad \text{right isometry: } QQ^\dagger = \mathbb{1}_m \end{aligned} \tag{3.4}$$

and we find that $\theta^\dagger = Q^\dagger L^\dagger$ is a QR, such that the two factorizations are essentially equivalent by swapping the roles of rows and columns.

3.1.3 Truncated QR-like decompositions (tQR)

Let us now focus on approximate low-rank factorizations and require only those properties that are actually required in tensor network methods. Consider, for example, the truncation step (2.35), where we require an approximate factorization $\theta \approx A \cdot C$, where the only relevant properties are the bounded dimension of the new index and the isometric property of the first factor. Due to the structural similarity to the QR decomposition, we call such factorizations truncated QR-like decomposition (tQR), captured by the following defining properties

$$\begin{aligned} &\theta \approx QR \qquad (\text{tQR}) \\ &\quad Q \in \mathbb{C}^{m\times k} \quad \text{left isometry: } Q^\dagger Q = \mathbb{1}_n \\ &\quad R \in \mathbb{C}^{k\times n} \quad \text{no required properties.} \end{aligned} \tag{3.5}$$

Note that unlike the QR, we do not impose a triangular structure on the second factor.

A decomposition of this form can be obtained from a truncated SVD as $\theta \approx U(SV^\dagger)$, with an optimal truncation error. Similarly, any dSVD, as introduced in section 3.1.4

below, yields a tQR. Relaxing to the weaker properties of the tQR allows us to also consider other truncated factorizations, such as QR decomposition with column pivoting (QRCP) [138, 139], see also [140, §5.4], or its randomized versions [141–143].

3.1.4 Deformed SVD

While truncated SVDs are commonly used for truncation in TNS algorithms, the property that the central factor $\tilde{S}$, the truncated singular values, are real, positive, diagonal is rarely needed. With MPS truncation, e.g. in (2.37), we only need it if we insist on a full canonical form. If we relax to an isometric form, however, we only require the isometric properties of the $\tilde{U}, \tilde{V}$ factors and that $\theta \approx \tilde{U}\tilde{S}\tilde{V}^\dagger$ is a good approximation. This motivates the following definition of a broader type of approximate factorization, which we dub deformed singular value decomposition (dSVD).

A *deformed* SVD of an $m \times n$ input matrix θ is an approximate factorization with the following properties

$$\begin{aligned} &\theta \approx U \Xi V^\dagger \qquad \text{(dSVD)} \\ &\qquad U \in \mathbb{C}^{m\times k} \qquad \text{left isometry: } U^\dagger U = \mathbb{1}_k \\ &\qquad \Xi \in \mathbb{C}^{k\times k} \qquad \text{no required properties} \\ &\qquad V \in \mathbb{C}^{n\times k} \qquad \text{left isometry: } V^\dagger V = \mathbb{1}_k \\ &\qquad \text{deformed singular value properties: } \theta V = U\Xi \text{ and } U^\dagger \theta = \Xi V^\dagger \ . \end{aligned} \tag{3.6}$$

We can understand it as arising from a truncated SVD, deformed by two independent unitary gauge transformations Q, P to the left and right of Ξ, that is, as

$$\theta \overset{\text{SVD}}{\approx} \tilde{U}\tilde{S}\tilde{V}^\dagger = (\tilde{U}Q)(Q^\dagger \tilde{S} P)(\tilde{V}P)^\dagger =: U \Xi V^\dagger. \tag{3.7}$$

The “deformed” singular value properties are an optional additional requirement. Note that they are trivially fulfilled for the approximation $\theta_{\text{approx}} := U\Xi V^\dagger$ but are a non-trivial requirement with the input matrix θ. They imply that the correction $\Delta = \theta - U\Xi V^\dagger$ is entirely in the orthogonal complements $U_\perp$ $(V_\perp)$ of U (V), meaning there is some $\Xi_\perp$ such that $\Delta = U_\perp \Xi_\perp V_\perp^\dagger$, and have consequences for AD, as discussed in section 4.2.2. In other words, the k columns of U (V) are linear combinations of only k left (right) singular vectors of θ. If the truncation is chosen such that those k singular vectors correspond to the k largest singular values of θ, the dSVD inherits the optimal truncation properties from the SVD.

Note that we do not require this to be the case and allow approximations with slightly larger truncation error $\|\theta - U\Xi V^\dagger\|$ than the optimal error achieved by a truncated SVD. A dSVD is only a sensible concept in the presence of truncation, that is if $k < \min(m, n)$, since e.g. for $k = m = n$, $\theta = \mathbb{1}\theta\mathbb{1}$ is a valid dSVD, but entirely useless.

By construction, a (truncated) regular SVD is a dSVD. It can also be achieved by postprocessing any truncated QR-like decomposition (tQR) $\theta \approx UM = ULQ$

with an additional LQ factorization. This approach is closely related to the QLP or UTV [144] factorizations, see e.g. [140, §5.4.6], which are also special cases of a dSVD, where the central factor Ξ has a triangular structure. More broadly, most common approximations to the SVD fulfill the requirements, such as e.g. randomized singular value decomposition (rSVD) [145] or randomized versions of the QRCP [141] or QLP [144, 146, 147]. The QR-based truncation routine introduced in section 3.2 is a dSVD as well.

3.2 QR-based truncation

The QR-based decomposition introduced in a previous publication [1] can be thought of as a dSVD $\theta \approx USV^\dagger$, that is as a subroutine that computes a low-rank factorization of a two-site wavefunction and can replace the truncated SVD in many settings. In particular, it can be used in the truncation step (2.37) of TEBD, which is the application highlighted in the publication. In this section, we rephrase the algorithm in a broader context and in the notation of this thesis.

The error

$$\epsilon = \left\lVert \theta - USV^\dagger \right\rVert_\mathrm{F} \tag{3.8}$$

is almost optimal in the following sense; The error of any rank k factorization is lower-bounded by the minimal error

$$\epsilon_\mathrm{SVD} = \sqrt{\sum_{i=k+1}^{\min(m,n)} \Lambda_i^2}, \tag{3.9}$$

where Λ_i are the singular values of θ in descending order, i.e. the smallest possible error is given by the weight of the discarded singular values and is achieved by a truncated SVD. The QR-based truncation is almost optimal, i.e. only slightly less accurate than a truncated SVD, in the sense that

$$\left| \epsilon - \epsilon_\mathrm{SVD} \right| \ll \epsilon_\mathrm{SVD}. \tag{3.10}$$

It can, therefore, replace the truncated SVD, which is typically used in MPS algorithms.

We emphasize that while we can derive heuristic explanations, the observation of small truncation errors is empirical. It should be explicitly verified in practice that the error ϵ is indeed small enough to be tolerable. Such a sanity check should be performed in tensor network simulations anyway to get a quantifiable handle on gauging if the bond dimension is large enough. Since U and $V^\dagger$ are not the exact singular vectors of θ, computing the error similarly to equation (3.9), i.e. based only on the discarded singular values in S is not possible and we need to explicitly evaluate equation (3.8). This comes at a non-negligible cost, with the same formal scaling as computing the QR-based truncation in the first place. Therefore, it may be beneficial to consider when the error is actually needed. In TEBD, for example, the error is not required at every time step; computing it at only every tenth time step or so allows the same kind of analysis in determining the maximum simulation time after which entanglement growth prohibits accurate MPS approximation.

The simple version of the QR-based truncation scheme, as described in [1, sec II], is rephrased in algorithm 3.1.

Algorithm 3.1 Simple QR-based Truncation

Given an $m \times n$ matrix θ and an $n \times k$ "initial guess" Ω, computes a dSVD $\theta \approx U \Xi V^\dagger$ with rank k, and properties listed in (3.6).

```
1. Form $Y = \theta\Omega$ and compute its QR decomposition $Y = UR$.
2. Form $\tilde{Y} = U^\dagger\theta$ and compute its LQ decomposition $\tilde{Y} = \Xi V^\dagger$.
```

We can think of $\Omega^\dagger$ as an initial guess for the right factor of θ and choose it in right isometric form if possible, making Ω itself left isometric. We propose to use the factor from the pre-update wavefunction $\hat{\theta} = \hat{E}\hat{F}$ as an initial guess, which is naturally given in right-isometric form in a TEBD simulation, that is, setting $\Omega = \hat{F}^\dagger$. In some settings, such as e.g. in a left sweep of DMRG, we may only have an isometric form established for the right factor $\hat{E}$. In that case, the QR-based truncation scheme can be readily adjusted by "vertically mirroring", e.g. by requiring an $m \times k$ matrix $\tilde{\Omega}$ and LQ decomposing $\tilde{\Omega}\theta$ instead of step 1, and doing a QR in step 2.

A heuristic explanation for why this algorithm achieves accurate truncation in the sense of (3.10) is to understand it as a variational single-site algorithm. Given an initial guess (E_i, F_i) for a factorization $\theta \approx EF$, let us assume that F_i is right isometric. Then, the optimal update for the first factor, which minimizes the distance $\|\theta - Y_i F_i\|_\mathrm{F}$, is given by $Y_i = \theta F_i^\dagger$. In order to make a similar update for the right factor, we shift the isometric form, using a QR decomposition, i.e. we transform our current best guess $(Y_i, F_i) \mapsto (Q_i, RF_i)$, which leaves the distance unchanged. Now the optimal right update is $\tilde{Y_i} = Q_i^\dagger\theta$ and we again shift the isometric form using an LQ decomposition $(Q_i, \tilde{Y_i}) \mapsto (Q_i L_i, \tilde{Q}_i) =: (E_{i+1}, F_{i+1})$, which concludes one sweep of single site updates. If we repeat this until convergence, e.g. for q steps, we effectively realized an alternating least squares (ALS) algorithm and have achieved $\theta \approx E_q F_q = Q_q L_q \tilde{Q}_q$, which already fulfills the target SVD-like isometric properties. This approach is transcribed in algorithm 3.2. An alternative perspective, why the QR-based truncation works is outlined in section 3.3.

Now, in the TEBD setting, we found that using the pre-update right factor $\hat{F}$ as an initial guess leads to convergence of the resulting error after only a single iteration, i.e., at $q = 1$. Setting $q = 1$ reduces algorithm 3.2 to algorithm 3.1. As observed empirically, it allows accurate truncation in TEBD time evolution, however, with a few minor drawbacks; Since the central bond matrix Ξ is not diagonal, we do not have direct access to the singular values of θ. Therefore, an MPS algorithm using the simple QR-based truncation scheme can not establish the full canonical form and has to work with an isometric form, with general (i.e. not diagonal) bond matrices in place of Schmidt values. Additionally, the new virtual space – the column space of $V^\dagger$ and row space of U – is fixed by the choice of Ω, e.g. to be the same as

Algorithm 3.2 Iterative Simple QR-based Truncation

Note: *This is included for pedagogical purposes only. In practice, we find that setting $q = 1$ is sufficient, such that the algorithm reduces to algorithm 3.1.*

Given an $m \times n$ matrix θ, an integer $q > 0$ and an $n \times k$ initial guess Ω, computes a dSVD $\theta \approx U \Xi V^\dagger$ with rank k, see (3.6).

```
1. Set $\tilde{Q}_0 = \Omega^\dagger$ and iterate the following steps for $i = 1, \dots, q$:
    1.a. Form $Y_i = \theta \tilde{Q}_{i-1}^\dagger$ and compute its QR decomposition $Y_i = Q_i R_i$.
    1.b. Form $\tilde{Y}_i = Q_i^\dagger \theta$ and compute its LQ decomposition $\tilde{Y}_i = L_i \tilde{Q}_i$.
2. Set $U = Q_q$, $\Xi = L_q$ and $V^\dagger = \tilde{Q}_q$.
```

before the time step in TEBD. This means that the bond dimension of the tensor network can not be adjusted dynamically; it can neither grow to accommodate growing entanglement nor can it be reduced if entanglement is still low early in the evolution. If symmetries are preserved, it additionally fixes the charge sectors of the new virtual leg, again to be the same as before for the time step, which severely limits the variational power of the ansatz, especially in transport simulation, where charges are expected to change significantly.

We can adjust the algorithm to address these shortcomings with the following two modifications; Firstly, by performing a truncated SVD of the bond matrix $L = \Xi$ as a final step. This allows us to select the optimal virtual space based on the singular values, however, constrained to a subspace of the column space of Ω. Secondly, by selecting an initial guess Ω with an enlarged virtual leg of dimension ℓ for the simple QR-based truncation, before truncating back to $k \leq \ell$ in the final step. Note that the $\ell \times \ell$ bond matrix L is smaller than the $m \times n$ input matrix θ, such that the cost of this final SVD is subdominant. The resulting scheme is given by algorithm 3.3 and achieves an almost optimal truncation, which allows dynamically adjusting the virtual space based on the singular values of L.

Algorithm 3.3 QR-based Truncation with bond expansion

Given an $m \times n$ matrix θ and an integer $\ell \leq \min(m, n)$, computes a dSVD $\theta \approx USV^\dagger$ with rank $k \leq \ell$, see (3.6).

```
1. Select a $\ell \times m$ column projection $\Pi$ and form the $n \times \ell$ test matrix $\Omega = (\Pi\theta)^\dagger$.
2. Form $Y = \theta\Omega$ and compute its QR decomposition $Y = QR$.
3. Form $\tilde{Y} = Q^\dagger \theta$ and compute its LQ decomposition $\tilde{Y} = L\tilde{Q}$.
4. Compute a dSVD of the $\ell \times \ell$ matrix $L \approx \tilde{U} S \tilde{V}^\dagger$.
5. Form $U = Q\tilde{U}$ and $V^\dagger = \tilde{V}^\dagger \tilde{Q}$.
```

This is the algorithm presented in [1, sec III], with variable names adjusted to facilitate discussion in the following sections. Additionally, the eigen-decomposition step is replaced with a dSVD; see the discussion in section 3.2.1. It remains to choose the entries of the projection Π. We propose to choose what we call a *column projection*, that is for an $\ell \times m$ projection Π we choose its rows from the rows of the identity matrix, such that $\Pi\theta$ is cheaply computed, simply by selecting rows from

θ. Here, rows of θ which are numerically close to zero should be avoided. In the TeNPy implementation, we choose the columns of θ with the largest vector 2-norms, though other choices are possible, such as randomly choosing from those columns with norms above a threshold.

An alternative is to start with the pre-update factor $\hat{F}$ and add rows of θ. This modification enables the intuition that $\Omega^\dagger$ is a good initial guess for a factor of θ, since

$$\theta \approx \hat{\theta} = \hat{E}\hat{F} = \begin{bmatrix}\hat{E} & 0\end{bmatrix} \begin{bmatrix}\hat{F} \\ \tilde{\Pi}\theta\end{bmatrix} =: \begin{bmatrix}\hat{E} & 0\end{bmatrix} \Omega^\dagger, \tag{3.11}$$

where the square brackets denote concatenation of matrices and $\tilde{\Pi}$ is an $(\ell - \hat{k}) \times m$ column projection. We also considered adding random rows instead of $\tilde{\Pi}\theta$. We found these variations to give virtually the same quality in terms of truncation error.

Similar to the simple QR-based truncation, we can understand steps 2 and 3 of algorithm 3.3 as single site updates that could be repeated (replacing Ω with $\tilde{Q}^\dagger$) until convergence. Again, we find empirically that a single iteration is sufficient to converge the truncation error.

The computational cost is dominated by the matrix products[1] in steps 2 and 3 and is in $\mathrm{O}(mn\ell)$. Truncation via standard SVD, for comparison, has a cost in $\mathrm{O}(mn \min(m, n))$. In the setting of an MPS update, we typically have $m = n = d\chi$, where d is the dimension of the physical on-site Hilbert space and χ the MPS bond dimension, which we assume to be uniform for simplicity. To keep the MPS bond dimension bounded, we want to truncate back to $k = \chi$. We found that in the setting of infinite time evolving block decimation (iTEBD), we can truncate the arising two-site wavefunctions with the same accuracy as SVD truncation if we choose $\ell \sim 1.1\chi$, i.e. ten percent larger than the target rank, and in particular independent of d. This results in a cost in $\mathrm{O}(d^2\ell\chi^2) = \mathrm{O}(d^2\chi^3)$ compared to the $\mathrm{O}(d^3\chi^3)$ for SVD truncation. Since the truncation is the dominant cost in TEBD, this results in a speedup factor of $\sim d$.

3.2.1 Decomposing the bond matrix

In the publication, we proposed to do an eigendecomposition of the hermitian square $L^\dagger L = \tilde{V}^\dagger S^2 \tilde{V}$ instead of an SVD of the bond matrix L, see step 4 of algorithm 3.3. As a result, the left singular vectors of L are not available, and we can not compute the (approximate) left singular vectors – the U factor in a dSVD (3.6) – of θ. This causes no issue in the context of TEBD, as they are not needed in (2.37). If U is needed, it could be computed as $\theta V S^{-1} \approx USV^\dagger V S^{-1} = U$, though the inverse singular values may be numerically unstable. Alternatively, if the algorithmic setting allows it, we can relax to a tQR, or rather an LQ version thereof, to be precise, by forming $\theta \approx (\theta V)V^\dagger$.

The reason for choosing to compute $S, \tilde{V}$ via this eigendecomposition in the publica-

[1] The QR and LQ decompositions have a cost in $\mathrm{O}(\ell^2 n)$ and $\mathrm{O}(\ell^2 m)$ respectively, and are assumed to be subdominant in the main text, as is the final SVD with cost in $\mathrm{O}(\ell^3)$.

tion is the performance on GPUs; the available GPU implementations for diagonalization of hermitian matrices are significantly faster than the SVD. The conditioning of the hermitian square is, however, generally worse than for the original matrix L, such that this eigendecomposition is less stable. This did not cause any problems for us, which we partially attribute to the preceding QR-based steps, which may already truncate the tail of the singular spectrum. Let us emphasize again that we trust the resulting factorization not because of these heuristics but because we observe that the truncation error – a quantity that should be analyzed anyway – is indeed small.

On CPU hardware, where SVD performance is comparable to hermitian diagonalization, decomposing the bond matrix L via SVD should always be preferred.

3.2.2 QR-based decomposition with symmetries

In the presence of symmetries, the input and output matrices have a block-sparse structure; see section 2.5. As a first version for doing QR-based truncation, we can consider simply applying algorithm 3.3 to each block θ_q of θ. This requires choosing a dimension ℓ_q for the projection Π *for each block*, which then upper bounds the number of singular values $k_q \leq \ell_q$ per block. While the total bond dimension $k = \sum_q k_q$ is typically prescribed in TNS context, it is unclear a priori how it should distribute among the charge sectors q for optimal truncation.

We propose a heuristic approach that uses an exactly factorized matrix $\hat{\theta} = \hat{E}\hat{F}$ that is close to θ and has the same block structure with sizes $\{m_q \times n_q\}$, e.g. from the tensor network before the update. Let the blockwise ranks of the factorization be $\hat{k}_q$; that is, the block sizes of $\hat{E}$ are $\{m_q \times \hat{k}_q\}$. The heuristic we propose is then to choose each ℓ_q slightly larger than the corresponding $\hat{k}_q$, up to the following two caveats; We can impose an upper bound $\ell_q \leq \min(m_q, n_q)$ since an exact decomposition is possible at equality. Further, we should impose a lower bound of $\mathrm{O}(1)$ to allow the update to explore new charge sectors, even if they are not present in the old leg. The resulting heuristic can be expressed as

$$\ell_q = \min\left[\max\left(\left\lceil(1+\rho)\hat{k}_q\right\rceil, \mu\right), m_q, n_q\right] \ , \tag{3.12}$$

where $\rho > 0$ is the *expansion rate*, $\mu \in \mathbb{N}_{>0}$ the *minimum block size* and $\lceil - \rceil$ denotes rounding to the next highest integer. We typically choose $\rho = 0.1$ and $\mu = 2$. This fully specifies the shape and block structure of the column projection Π; it remains to choose the entries. Again, vanishing columns of θ should be avoided, and we propose to select only columns with vector norm above a threshold or pick the columns with largest norms *in each block*.

As in the non-symmetric case, the test matrix can be modified to use the rows of $\hat{F}$, supplemented with either rows of θ or random vectors. This modification would mean choosing the blocks of Ω as

$$\Omega_q = \left[\hat{F}_q^\dagger \quad (\tilde{\Pi}_q \theta_q)^\dagger\right] \ , \tag{3.13}$$

where $\tilde{\Pi}_q$ is a $(\ell_q - \hat{k}_q) \times m_q$ column projection.

As an improvement over a strictly blockwise decomposition, the SVD in step 4 should be truncated by keeping the largest singular values overall, i.e. coordinating which singular values to keep among all blocks, as discussed in section 2.5. The resulting algorithm for a QR-based decomposition is outlined in algorithm 3.4. It is implemented[2] in TeNPy.

Algorithm 3.4 QR-based truncation for block-sparse matrices

Given an $m \times n$ block-sparse matrix θ with block sizes $\{m_q \times n_q\}$ and integers $\ell_q \leq \min(m_q, n_q)$ for every block q, computes an approximate block-sparse dSVD $\theta \approx USV^\dagger$ with block-wise ranks $k_q \leq \ell_q$.

```
1. Select a block-sparse column projection $\Pi$ with block sizes $\{\ell_q \times m_q\}$.
2. Form $\Omega = (\Pi\theta)^\dagger$ by block-wise matrix product.
3. Form $Y = \theta\Omega$ and compute its block-wise QR decomposition $Y = QR$.
4. Form $\tilde{Y} = Q^\dagger\theta$ and compute its block-wise LQ decomposition $\tilde{Y} = L\tilde{Q}$.
5. Compute a (truncated) SVD: $L \approx \tilde{U}S\tilde{V}^\dagger$, following algorithm 2.8.
6. Form $U = Q\tilde{U}$ and $V^\dagger = \tilde{V}^\dagger\tilde{Q}$ by block-wise matrix product.
```

3.3 Connection to randomized linear algebra

We became aware after publication that the QR-based truncation algorithm is closely related to the ideas of randomized linear algebra [148]. In this section, we highlight these parallels and differences, and put the ideas of QR-based truncation into the framework of randomized matrix factorization.

Let us first summarize the setup and general idea for randomized matrix decompositions. The input is an $m \times n$ matrix θ with numerical rank k, meaning it has rank k up to floating point errors, i.e. only its first k singular values are distinguishable from zero at machine precision.

The first step in a randomized factorization is to obtain a $m \times \ell$ left isometry Q that approximates the range of θ, such that $QQ^\dagger\theta \approx \theta$. If $\ell = k$, this is guaranteed to be possible, however numerically just as hard as simply computing a standard factorization of θ, e.g. truncated SVD. The idea is then to allow oversampling, i.e. allow ℓ slightly larger than k. This typically involves drawing a random test matrix, which we can understand as ℓ sample vectors from some distribution. For a fixed sample size ℓ, error estimates, i.e. bounds on the truncation error $\epsilon_Q = \|(\mathbb{1} - QQ^\dagger)\theta\|_\mathrm{F}$ can be proven to hold with some fixed probability close to one.

As a result, an approximate factorization $\theta \approx Q(Q^\dagger\theta)$ with rank ℓ is achieved. As a second step, it can be post-processed to further reduce the rank to k or achieve

[2] The truncation is implemented as `tenpy.algorithms.truncation.decompose_theta_qr_based` and is used by `tenpy.algorithms.tebd.QRBasedTEBDEngine` realizing TEBD, as well as by `tenpy.algorithms.mps_common.QRBasedVariationalApplyMPO` realizing variational MPO application.

desirable properties such as those of a dSVD (3.6). If the numerical rank k and thus the sample size ℓ are small compared to the input size m, n, this can be done by acting with standard factorizations on smaller matrices, e.g. on $Q^\dagger\theta$, and is thus cheaper than direct standard factorizations of θ. This concludes a brief overview that is, of course, condensed and simplistic, omitting many possible algorithmic improvements and variations.

A common randomized algorithm for computing an approximate truncated factorization uses a structured test matrix Ω such that the matrix product $\theta\Omega$ can be computed efficiently. The test matrix proposed by Halko et al in [148, chpt. 4] is given by $\Omega = \sqrt{n/\ell}DF\Pi$, where D is a $n \times n$ diagonal matrix of uniformly distributed random complex phases, F is the $n \times n$ discrete Fourier transform and Π is a random $n \times \ell$ column projections, whose rows are randomly selected from the rows of the $n \times n$ identity matrix. Therefore, $Y = \theta\Omega$ can be computed by applying the phases from D scaled by $\sqrt{n/\ell}$, followed by a subsampled fast Fourier transform (FFT) [149] at a total cost in $\mathrm{O}(mn\log\ell)$. They refer to this class of test matrix Ω as subsampled random Fourier transforms (SRFTs).

Algorithm 3.5 Randomized SVD with Fast Randomized Range Finder

This is an equivalent reframing of algorithms 4.5 and 5.1 in reference [148] to match the notation of this thesis.

Given an $m \times n$ matrix θ and an integer $\ell \leq \min(m, n)$, computes an approximate SVD $\theta \approx USV^\dagger$ with rank $k \leq \ell$.

```
1. Draw a $n \times \ell$ random SRFT (see main text) test matrix $\Omega$.
2. Form $Y = \theta\Omega$ via subsampled FFT
3. Compute the QR decomposition $Y = QR$.
4. Form the $\ell \times n$ matrix $B = Q^\dagger\theta$ and and compute its SVD $B = \tilde{U}SV^\dagger$.
5. Form $U = Q\tilde{U}$.
```

The resulting algorithm, the randomized SVD, is transcribed in algorithm 3.5. Steps 1-3 realize a randomized range finder, and steps 4-5 can be thought of as post-processing to achieve the SVD properties.

As a first notable difference, for the QR-based truncation – and in the TNS context in general – we do *not* assume that we truncate only to the numerical rank k of the matrix, but to some pre-determined target rank k that is potentially significantly smaller. While we require the truncation error (3.8) to remain small, for the simulation to be sensible, the resulting rank k may be significantly smaller than the numerical rank of the input matrix θ, i.e. we *do* discard singular values, that – while "small" – are significantly larger than machine precision.

Let us then compare the QR-based truncation variants of the previous section with the randomized scheme of algorithm 3.5. Compared with the simple QR-based truncation of algorithm 3.1, we can identify a similar broad structure but see a different test matrix Ω; instead of a random sample, it uses an informed initial guess. The following QR step is the same as for the randomized SVD algorithm and concludes the range finder; as we confirm empirically, we have $UU^\dagger\theta \approx \theta$ at this point. In addition to the heuristic of the previous section, we can understand this result also in the context of the randomized range finder; the deterministically selected test matrix of the QR-based scheme seems to be typical enough of a random distribution to allow the same accurate approximation of the range of θ. In other words, the choice of the test matrix is not fine-tuned enough to hit those low-probability cases where the randomized range finder breaks down. The second step of 3.1 is then simply post-processing to the target SVD-like factorization. The pedagogical example of iterating single site updates, as outlined in algorithm 3.2 can be understood as an alternative randomized range finder, the "randomized subspace iteration", see [148, alg. 4.4], which realizes a power method to improve the subspace spanned by Ω. As a simple version with no subsequent further truncation, the simple scheme has $\ell = k$.

The full QR-based truncation scheme of algorithm 3.3, on the other hand, does allow for oversampling with $\ell > k$. Other than that, the range-finding step is the same as before and can be understood in the same way. The post-processing, realized in steps 3-5, on the other hand, is unnecessarily complicated and exists in this form since it arose incrementally from the simple algorithm 3.1. Indeed, we can instead compute a truncated SVD directly of $\tilde{Y} = \tilde{U}SV^\dagger$, omitting the LQ factorization, as is done in the randomized SVD. The resulting cost has the same formal scaling but with a smaller prefactor.

3.4 Synthesized truncation routines

In this section, we propose combined routines, incorporating the lessons learned from comparison with randomized linear algebra into the QR-based truncation routines.

We keep the two-stage structure of a range-finding stage and a post-processing stage. First, for the test matrix Ω in the range finder, we propose to use rows from the pre-update factor[3] $\hat{F}$, and add additional samples from SRFT to achieve the target oversampled rank ℓ, that is choosing

$$\Omega = \begin{bmatrix} \hat{F}^\dagger & \sqrt{\frac{n}{\ell - \hat{k}}} D F \tilde{\Pi} \end{bmatrix} \ , \tag{3.14}$$

where, as in the previous section, the second block can be computed via subsampled FFT.

After the range finder stage, we found a Q such that $\theta \approx Q(Q^\dagger\theta)$, and perform a standard factorization of $Y = Q^\dagger\theta$ next. There are approaches to avoid forming the explicit matrix product that defines Y to bring down the computational cost, losing some accuracy as a trade-off, as outlined in reference [148, section 5.2]. In a TNS context, this is typically not worth it, as the tensor contractions in the rest of the algorithm – outside the truncation subroutine – have a cost with the same or higher cost scaling than forming Y anyway. In TEBD, for example, forming the two-site

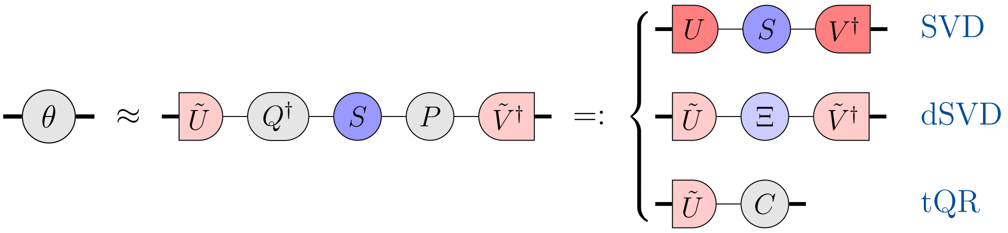
[3] Recall that we assume that we have access to a related matrix $\hat{\theta} \approx \theta$ with an exact factorization $\hat{\theta} = \hat{E}\hat{F}$ with rank $\hat{k} \leq k$.

wavefunction θ in the first place has the same formal scaling in $\mathrm{O}(d^2\chi^3)$ as forming Y explicitly.

The straightforward choice for the standard factorization of Y is an SVD. Having GPU acceleration in mind, however, we propose to investigate other choices, which result in a factorization of θ with weaker properties. In particular, we consider as alternatives the deformed singular value decomposition (dSVD) or tQR, see sections 3.1.4 and 3.1.3.

Any of these factorizations can be post-processed into any other. If given an SVD $\theta_{\text{approx}} = USV^\dagger$, we already have a dSVD trivially and $\theta_{\text{approx}} = U(SV^\dagger)$ is a tQR. Given a dSVD $\theta_{\text{approx}} = \tilde{U}\tilde{S}\tilde{V}^\dagger$, we can perform an SVD of $\tilde{S} = U'S(V')^\dagger$, to obtain the SVD $\theta_{\text{approx}} = (\tilde{U}U')S(\tilde{V}V')^\dagger$, or form the tQR $\theta_{\text{approx}} = \tilde{U}(\tilde{S}\tilde{V}^\dagger)$. Given a tQR $\theta_{\text{approx}} = QR$, we can perform an SVD $R = U'SV^\dagger$ of R, to obtain the SVD $\theta_{\text{approx}} = (QU')SV^\dagger$, or if we just want a dSVD, it is enough to perform either a QR or tQR of $R^\dagger = Q'\tilde{S}'$ to obtain the dSVD $\theta_{\text{approx}} = QS'^\dagger Q'^\dagger$.

We can understand the relationship between these different forms of factorization in the following way;



(3.15)

The inner legs describe the fine-tuned basis choice of the singular vectors of θ, such that the central factor S is real, non-negative, and diagonal. Then, the unitary basis transformations Q, P map to some other basis of the same respective subspaces, such that the combined central matrix $\Xi := Q^\dagger SP$ no longer has any particular properties. Since the bases span the same spaces, however, any optimal truncation properties are inherited from the SVD. We can combine the factors in different ways as indicated to obtain the different types of factorizations.

Going from one factorization to the other is cheap compared to forming them in the first place, as the procedures above act on matrices that have the smaller truncated dimension $k < min(m, n)$ as either width or height. Nevertheless, we emphasize that it is worth it to reconsider in the TNS algorithm, which properties are actually needed and meet the truncation routine halfway.

Finally, to post-process $\theta \approx QY$ to any of the three classes of factorizations (SVD, dSVD and tQR), we need to perform the same kind of decomposition of Y and absorb its left factor into Q.

3.5 Benchmark

For a benchmark of the QR-based truncation routine, we consider the d-state quantum clock model

$$H = -\sum_n \left(Z_n Z_{n+1}^\dagger + \text{h.c.} \right) - g \sum_n \left(X_n + \text{h.c.} \right) \ , \tag{3.16}$$

where the clock operators are given by

$$Z = \begin{pmatrix} 1 & & & & \\ & \omega & & & \\ & & \omega^2 & & \\ & & & \ddots & \\ & & & & \omega^{d-1} \end{pmatrix} , \ X = \begin{pmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & 0 & \ddots & \\ & & & \ddots & 1 \\ 1 & & & & 0 \end{pmatrix} , \tag{3.17}$$

with $\omega = \mathrm{e}^{2\pi \mathrm{i}/d}$. We consider the model on an infinite chain. The model can be seen as a generalization of the transverse field Ising model (TFIM), which is the $d = 2$ special case. It is particularly suited to highlight the scaling with the dimension d of the on-site Hilbert space. The model has a critical point at $g = 1$ for $d \leq 4$ and an extended critical region for $d \geq 5$ [150, 151].

We start with the $Z = 1$ product state and evolve it in time with the $g = 2$ Hamiltonian. This constitutes a global quench from $g = 0$ to $g = 2$, crossing at least one critical point. We perform the simulation using the iTEBD algorithm, with updates given by (2.36) in four algorithmic variations. First ("SVD"), we use a standard truncated SVD $\tilde{\theta} \approx USV^\dagger$ for decomposing the evolved wavefunction. Secondly ("EIG"), we perform the same decomposition but numerically evaluate it by diagonalizing the hermitian square $\tilde{\theta}^\dagger \tilde{\theta} = VS^2V^\dagger$, see the discussion in section 3.2.1. Thirdly ("QR"), we employ the simple QR-based truncation routine of algorithm 3.1 and lastly ("QR+CBE"), we use the QR-based truncation with bond expansion, as described in algorithm 3.3. We run the benchmark on an NVIDIA A100 GPU (80GB RAM) with CUDA version 11.7, as well as an AMD EPYC 7763 CPU with 64 physical cores and MKL version 2019.0.5. The two units have similar power consumption: 300W and 280W thermal design power, respectively. All simulations are performed in double precision (i.e., `complex128` in python). The implementation used for the benchmark and the data are available on GitHub[4].

In Fig. 3.1, we perform full TEBD simulations of the quench protocol for a $d = 5$ clock model. We run the simulation beyond times where the approximation of the evolved state as an MPS of the given bond dimension breaks down, as quantified by a large truncation error. In the time regime of acceptable error $\epsilon_{\text{trunc}} \lesssim 10^{-5}$, that is until $t \lesssim 2$ depending on bond dimension, we observe excellent agreement between the different TEBD schemes in the extracted expectation values $\langle Z \rangle$ and entanglement entropy S_{vN} up to relative deviations of $10^{-11} \sim 10^{-12}$. For the QR-based scheme, we do not have access to all singular values of $\tilde{\theta}$, from which the

[4] https://github.com/Jakob-Unfried/Fast-Time-Evolution-of-MPS-using-QR

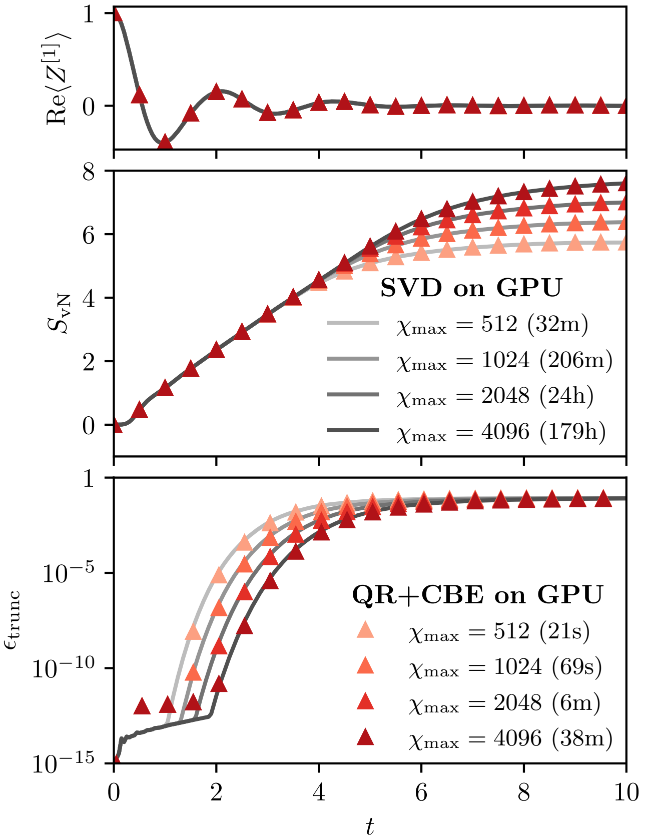


Figure 3.1: TEBD Simulation of a global quench in the $d = 5$ quantum clock model from $g = 0$ to $g = 2$ with a time step of $\delta t = 0.05$. We show a local Z expectation value (top), the half-chain von Neumann entanglement entropy (center), and truncation error (bottom). We compare data from SVD-based (solid lines) and QR-based (triangles) TEBD simulations at a range of bond dimensions $\chi_{\max}$ (colors). For the QR-based scheme, we employ controlled bond expansion, that is algorithm 3.3, with $\ell = \max(100, 1.1\chi)$ and plot only every tenth data point. For both schemes, we discard Schmidt values smaller than 10^{-14} and keep at most $\chi_{\max}$ of them. Time in the legend denotes the total wall time needed for each simulation, i.e. to generate the shown data from scratch.

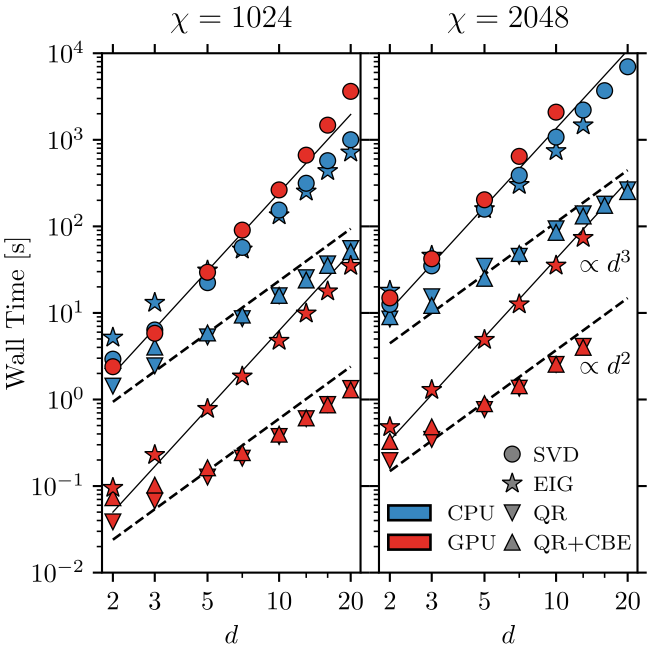


Figure 3.2: Timing benchmark for the application of a single gate to an MPS for different hardware (marker colors) and truncation schemes (marker shapes). We give the average wall time needed to perform a single TEBD update, that is contracting and decomposing the evolved wavefunction θ given by (2.36) and contracting the new B tensor according to (2.37). For the decomposition of $\tilde{\theta}$, we consider the following different schemes; a (truncated) SVD, a truncated hermitian eigendecomposition $\tilde{\theta}^\dagger\tilde{\theta} \approx V^\dagger S^2 V$, the simple QR-based truncation described in algorithm 3.1, and the QR-based truncation with bond expansion (CBE) described in algorithm 3.3. For CBE, we replace the truncated SVD of the bond matrix with a truncated hermitian eigendecomposition and choose $\ell = 1.1\chi$, the same expansion rate as for Fig. 3.1. The initial MPS has a bond dimension χ, and the evolved state is truncated to the same dimension χ. Solid (dashed) lines are powerlaws with the expected cubic (quadratic) scaling with the physical dimension d. The missing data points for large d in the right panel were not possible to obtain on the available hardware due to memory limitations.

truncation error is extracted in SVD-based TEBD. We instead explicitly compute the distance between the evolved wave function $\tilde{\theta}$ and its low-rank approximation.

In Fig. 3.2, we benchmark runtimes for the core algorithmic step (2.36) of contracting and subsequently decomposing the evolved wavefunction $\tilde{\theta}$, that is evaluating equation (2.36) and (2.37). We repeat this for all combinations of truncation scheme and hardware, as well as a range of Hilbert space dimensions d.

We clearly observe the improved scaling of the QR-based algorithm, which is quadratic in d instead of cubic, as well as a speed-up of one to two orders of magnitude from hardware acceleration for EIG and QR-based algorithms. For example, the QR-based truncation scheme on the GPU with $\chi = 1024$, $d = 20$ reaches a speed-up factor of 2700 compared to the SVD-based scheme on the same GPU and 750 compared to SVD on CPU.

3.6 Conclusion

In this chapter, we have established the properties of truncated factorizations that are actually required by common tensor network algorithms, e.g. in terms of the deformed singular value decomposition (dSVD) for MPS algorithms in isometric form. We have proposed the QR-based truncation routine, which is a dSVD, discussed its relation to randomized numerical linear algebra, and suggested a best-of-both-worlds synthesis. We have demonstrated that the QR-based truncation scheme allows simulation of the time evolution of MPS to the same degree of accuracy, but compared to the SVD-based scheme drastically increases runtime, especially on GPU hardware. The improved scaling with the local Hilbert space dimension d implies substantial performance increase even on CPU for large d, e.g. in simulations of open systems or bosonic systems.

The truncation schemes can be used to accelerate TNS truncation in a broader class of algorithmic settings. This has already been successful for MPO evolution in Ref. [152]. Applications of the randomized SVD have been suggested in [153], but to our knowledge, they have not been systematically compared in a published work. Studying the interaction with the decompositions proposed in this chapter with DMRG, and in particular subspace expansion approaches, is another interesting avenue for future development. The algorithm should seamlessly apply to truncation steps in isometric TNS [70, 71] as well.

To our knowledge, it is an open question how to compute approximations to the SVD, such as e.g. the QLP decomposition (QLP) for symmetric matrices. This is because determining the target ranks for each block is challenging without access to the full singular value spectrum.

We also suggest a full comparison of different methods for future work. This should compare (a) the different range finders featured in Algorithm 3.5, in Algorithm 3.1, the synthesized version in equation (3.14), or not performing randomized range finding at all and directly decomposing the full matrix. Next (b), it should compare the dense matrix factorization steps, covering e.g. the standard SVD, as well as the QRCP and QLP. Lastly, it should run (c) and a wider class of different hardware and (d) consider conservation of symmetries versus not doing so.

Chapter 4

Gradient-based optimization of Projected Entangled Pair States (PEPS)

The variational methods for PEPS optimization based on local updates have not been able to fulfill the promise of extending the success of DMRG to TPS in higher dimensions. We attribute this in part to the lack of a canonical form with orthogonality centers. Recent developments of gradient-based methods [37, 38, 129, 154], and in particular, approaches using automatic differentiation (AD) have found success in optimizing PEPS for infinite systems. Applying similar methods to finite PEPS, however, seems to cause problems in the stability of the resulting algorithms and require ad-hoc adjustments [115, 125]. We attribute this to the interaction of the gradient-based approach with the approximations that are necessary to evaluate the loss function being optimized, e.g. the variational energy.

In this chapter, we propose a scheme for evaluating the gradients that results in stable optimization trajectories, and derive from it a ground state search and a time evolution algorithm for finite PEPS. We introduce the gradient-based approaches in section 4.1, formulating both ground state search and time evolution as optimization problems. We discuss automatic differentiation (AD) as an approach to compute the gradients of the resulting cost functions in 4.2, and then focus on the explicit gradient evaluation scheme in section 4.3. We show benchmark results of the resulting algorithms in section 4.4 before concluding in section 4.5.

4.1 Gradient based approach

A gradient-based optimization method is a global method, where *simultaneous* updates of all tensors are derived based on values and gradients of some target loss function. For ground state search, this means finding the PEPS tensors $\{A^{[x,y]}\}$

parametrizing a trial state $|\phi\rangle$ such that the variational energy

$$E\big(\,|\phi\rangle\,\big) := \frac{\langle\phi|H|\phi\rangle}{\langle\phi|\phi\rangle} \tag{4.1}$$

as a function of the PEPS tensors $\{A^{[x,y]}\}$ is minimized. This gives us a ground state approximation

$$|\mathrm{GS}\rangle \approx \mathop{\arg\min}_{|\phi\rangle\in\mathrm{PEPS}(D)} E\big(\,|\phi\rangle\,\big). \tag{4.2}$$

Equation (4.2) becomes exact as $D\to\infty$, and at a given finite D, we obtain a PEPS approximation using a numerical optimization algorithm, such as conjugate gradient or quasi-Newton methods. This requires us to evaluate the loss function and its gradient at any point in the variational manifold. The "point" is given by a set $\{A^{[x,y]}\}$ of tensors that parametrize a PEPS $|\phi\rangle$ and the gradient is similarly a set of components

$$G^{[x,y]} := \frac{\partial E(|\phi\rangle)}{\partial \overline{A}^{[x,y]}}. \tag{4.3}$$

As a result, we get a gradient-based ground state search, as e.g. employed in Refs. [38, 115].

We propose a time evolution method using a similar gradient-based minimization of the square distance between the exactly evolved state and a trial state of bounded bond dimension. This has the same goal as the MPO evolution algorithm for MPS time evolution but is approached with global gradient-based updates instead of local variational updates. Assume we have an approximation of the unitary time evolution operator $U(\delta t)$ for a small time step in the form of a PEPO. Now, by minimizing the square distance

$$\begin{aligned}\Delta^2\big(\,|\phi\rangle\,, U(\delta t)\,|\psi(t)\rangle\,\big) &:= \left\| \frac{|\phi\rangle}{\||\phi\rangle\|} - \frac{U(\delta t)|\psi(t)\rangle}{\|U(\delta t)|\psi(t)\rangle\|} \right\| \\ &= 2 - 2\frac{\mathrm{Re}\langle\phi|U(\delta t)|\psi(t)\rangle}{\sqrt{\langle\phi|\phi\rangle\langle\psi(t)|\psi(t)\rangle}}\end{aligned} \tag{4.4}$$

we obtain an approximation of the evolved state up to normalization. Concretely, this means

$$\frac{1}{\mathcal{N}}\,|\psi(t+\delta t)\rangle = \frac{1}{\mathcal{N}} U(\delta t)\,|\psi(t)\rangle \approx \mathop{\arg\min}_{|\phi\rangle\in\mathrm{PEPS}(D)} \Delta^2\big(\,|\phi\rangle\,, U(\delta t)\,|\psi(t)\rangle\,\big) \tag{4.5}$$

with some normalization factor $\mathcal{N} > 0$. Again, the approximation becomes exact if the bond dimension $D\to\infty$ is unbounded, such that the minimization explores the entire many-body Hilbert space.

The time evolution method can be generalized to approximately apply arbitrary operators, that is, optimize $|\phi^\star\rangle \approx \frac{1}{\mathcal{N}} O|\psi\rangle$ for a PEPO O and a PEPS $|\psi\rangle$. Note, however, that we used the unitarity of the time evolution operator in equation (4.4), and in the general case with non-unitary O, we have

$$\Delta^2\big(\,|\phi\rangle\,, O\,|\psi\rangle\,\big) = 2 - \frac{2}{\|O|\psi\rangle\|}\frac{\mathrm{Re}\langle\phi|O|\psi\rangle}{\sqrt{\langle\phi|\phi\rangle}} =: 2 - \frac{2}{\|O|\psi\rangle\|}\Omega(|\phi\rangle\,, O\,|\psi\rangle). \tag{4.6}$$

Evaluating the norm $\sqrt{\langle\psi|O^\dagger O|\psi\rangle}$ in the denominator may be prohibitively expensive, and since it is constant as a function of the trial state $|\phi\rangle$, we may instead maximize the overlap Ω. Since Δ^2 is a strictly decreasing function of Ω, this is equivalent and the only downside is that unlike $\Delta^2 = 0$, we do not know the theoretical optimum value for Ω at which the approximation $|\phi^\star\rangle \approx \frac{1}{N}O|\psi\rangle$ becomes exact.

The remaining challenge is evaluating the gradients, e.g. (4.3) of the variational energy, and similarly the gradients of (4.4).

4.2 Gradients from automatic differentiation

A popular approach is to compute the gradients using automatic differentiation (AD), which can take the implementation of a function and compute its derivative by composing known derivative formulae of its building blocks using the chain rule.

4.2.1 Automatic differentiation – a brief introduction

We refer to Refs. [155, 156] for detailed introductions to and reviews of AD. Let us establish some terminology and give a simple example for reverse-mode AD. The central objects in reverse-mode are the adjoints or cotangents Γ_X associated with every variable X. They can be thought of as derivatives $\Gamma_X = \partial\mathcal{L}/\partial X$ of the target loss function $\mathcal{L}$ such that $\mathrm{d}\mathcal{L} = \sum_i \Gamma_{X_i}\mathrm{d}X_i$ in the real case. In the complex case (with a real-valued cost function $\mathcal{L}$ of complex variables X_i), we can think of X_i and $\overline{X}_i$ as independent variables and have

$$\mathrm{d}\mathcal{L} = \sum_i \left(\Gamma_{X_i}\mathrm{d}X_i + \Gamma_{\overline{X}_i}\mathrm{d}\overline{X}_i\right) = \sum_i \left(\Gamma_{\overline{X}_i}\mathrm{d}\overline{X}_i + \text{c.c.}\right) \ . \tag{4.7}$$

In most AD schemes for complex variables, only the adjoints of the conjugate variables, that is, the $\Gamma_{\overline{X}_i}$ are stored, since the $\Gamma_{X_i} = \overline{\Gamma_{\overline{X}_i}}$ are not independent. Now, for a matrix A of variables A_{ij}, this takes the convenient form

$$\mathrm{d}\mathcal{L} = \sum_{ij}\left(\Gamma_{\overline{A}_{ij}}\mathrm{d}\overline{A}_{ij} + \text{c.c.}\right) = \mathrm{Tr}\left(\Gamma_{\overline{A}}\mathrm{d}A^\dagger + \text{c.c.}\right) \ . \tag{4.8}$$

As a concrete example, let us derive the AD formula for a basic function in the real case. Consider addition of two variables, that is forming $x = a + b$. The goal is now to express the input cotangents Γ_a, Γ_b in terms of the output cotangent Γ_x, as well as the values x, a, b. To this end, equate $\Gamma_x\mathrm{d}x = \mathrm{d}\mathcal{L} = \Gamma_a\mathrm{d}a + \Gamma_b\mathrm{d}b$ and plug in the differential $\mathrm{d}x = \mathrm{d}a + \mathrm{d}b$ of the defining equation to find $\Gamma_a = \Gamma_b = \Gamma_x$. Similarly, for taking a power $y = c^n$, we find $\Gamma_c = nc^{n-1}\Gamma_y$.

The core of automatic differentiation is then to derive the input cotangents of a composite function $\mathcal{L}$, given these kinds of formulae for a set of building block functions that $\mathcal{L}$ is composed of. In reverse-mode, this is done by establishing a computation graph for the function $\mathcal{L}$, which formally assigns distinct variable names to each intermediate result in the computation of a value of $\mathcal{L}$ and organizes the order in which they are computed in a graph. The computation of a value of $\mathcal{L}$ is commonly referred to as the forward pass. It starts with values for the input variables and terminates with a value for the output $\mathcal{L}$. This graph is then traversed in the opposite direction, in the "backward pass", starting from the cotangent $\Gamma_{\mathcal{L}} := 1$, and terminates with cotangents of the input variables.

As a concrete example, consider the function $\mathcal{L}(x, y) = (4x + y)^2$ with input values $x = 1/2$ and $y = 3$. For the forward pass we compute

$$\begin{aligned} w_1 &:= 4x = 2 \\ w_2 &:= w_1 + y = 5 \\ \mathcal{L} &= (w_2)^2 = 25 \end{aligned} \tag{4.9}$$

and for the backward pass we start with $\Gamma_{\mathcal{L}} := 1$ and compute

$$\begin{aligned} &\Gamma_{w_2} = 2w_2\Gamma_{\mathcal{L}} = 10 \\ &\Gamma_{w_1} = \Gamma_{w_2} = 10 \quad ; \quad \Gamma_y = \Gamma_{w_2} = 10 \\ &\Gamma_x = 4\Gamma_{w_1} = 40 \ . \end{aligned} \tag{4.10}$$

This computation has given us the derivatives $\partial\mathcal{L}/\partial x = \Gamma_x = 40$ and $\partial\mathcal{L}/\partial y = \Gamma_y = 10$, evaluated at $(x, y) = (1/2, 3)$, which we can easily verify by hand. It results from going through the steps of (4.9) in reverse order and applying the AD formula for each respective operation. In particular, this can be automated, resulting in automatic differentiation (AD).

4.2.2 Backward formula for the truncated SVD

It is crucial, and not widely implemented in common AD libraries, to use appropriate backward formulae for the truncated SVD. The backward formula for the complex SVD was only found recently [157], and the use of AD in the context of PEPS optimization has recently led to the introduction of corrected formulae in the presence of truncation [39]. We state the backward formula for the truncated SVD here, and refer to the derivations, and related result for truncated hermitian eigendecompositions in appendix B. The result for the general case is essentially a restatement of the results of Ref. [39], and we develop a new result for simplification in the special case of the enlarged gauge transformation, where requirements on the decomposition are relaxed to those of a deformed singular value decomposition (dSVD).

For the purposes of AD, we view the truncated SVD is a mapping $A \mapsto (U, S, V)$ of an input matrix $A \in \mathbb{C}^{m\times n}$ such that

$$A = USV^\dagger + XYZ^\dagger. \tag{4.11}$$

Here, $U \in \mathbb{C}^{m\times k}$, $X \in \mathbb{C}^{m\times(m-k)}$, $V \in \mathbb{C}^{n\times k}$ and $Z \in \mathbb{C}^{n\times(n-k)}$ are (left) isometries and X (Z) is the orthogonal complement of U (V) and $S \in \mathbb{R}^{k\times k}$ and $Y \in$

$\mathbb{R}^{(m-k)\times(n-k)}$ are real matrices that vanish off the main diagonal, where S is strictly positive and Y is non-negative. We consider $A \approx USV^\dagger$ as the approximation, the truncated SVD. The result for the AD backward formula for the function $A \mapsto (U, S, V)$, in particular with a non-conjugated V as the third output is

$$\begin{aligned}\Gamma_{\overline{A}} &= \Gamma^{(\mathrm{S})}_{\overline{A}} + \Gamma^{(\mathrm{Uo})}_{\overline{A}} + \Gamma^{(\mathrm{Vo})}_{\overline{A}} + \Gamma^{(\mathrm{diag})}_{\overline{A}} + \Gamma^{(\mathrm{tr})}_{\overline{A}} \\ &= \frac{1}{2}U\left(\Gamma_{\overline{S}} + \Gamma^\dagger_{\overline{S}}\right)V^\dagger + U(J+J^\dagger)SV^\dagger + US(K+K^\dagger)V^\dagger \\ &\quad + \frac{1}{2}US^{-1}(L^\dagger - L)V^\dagger + \left[XX^\dagger\gamma V^\dagger + U\varphi^\dagger ZZ^\dagger\right] ,\end{aligned} \tag{4.12}$$

where

$$J := F \circ (U^\dagger\Gamma_{\overline{U}}) \quad ; \quad K := F \circ (V^\dagger\Gamma_{\overline{V}}) \quad ; \quad L := \mathbb{1} \circ (V^\dagger\Gamma_{\overline{V}}). \tag{4.13}$$

$$F_{ij} := \begin{cases} 0 & i = j \\ 1/(S_i^2 - S_j^2) & i \neq j \end{cases} \tag{4.14}$$

Here, $\circ$ denotes elementwise matrix multiplication and γ, φ are the solutions to the coupled Sylvester equations

$$\begin{aligned}\Gamma_{\overline{U}} &= \gamma S - AZZ^\dagger\varphi \\ \Gamma_{\overline{V}} &= \varphi S - A^\dagger XX^\dagger\gamma \ .\end{aligned} \tag{4.15}$$

The Sylvester equations (4.15) have a unique solution if and only if S and Y have no singular values in common, such that splitting multiplets of (nearly) degenerate singular values should be avoided. The formula simplifies in the following relevant special cases:

1. For a real SVD, i.e. such that the input A and all outputs U, S, V are real-valued, we find $\Gamma^{(\mathrm{diag})}_{\overline{A}} = 0$.

2. If there is no truncation, i.e. if $k = \min(m, n)$, the Sylvester equations have a closed form solution which results in $\Gamma^{\mathrm{tr}}_{\overline{A}} = XX^\dagger\Gamma_{\overline{U}}S^{-1}V^\dagger + US^{-1}\Gamma^\dagger_{\overline{V}}ZZ^\dagger$.

3. If A is square and there is no truncation, that is for $k = m = n$, we have $\Gamma^{\mathrm{tr}}_{\overline{A}} = 0$.

4. If the loss function is invariant under the enlarged gauge transformation

$$U \mapsto UQ \quad , \quad S \mapsto Q^\dagger SR \quad , \quad V \mapsto VR \tag{4.16}$$

with arbitrary $k \times k$ unitaries Q, R, that is if $\mathcal{L}(U, S, V) = \mathcal{L}(UQ, Q^\dagger SR, VR)$, we have $\Gamma^{(\mathrm{Uo})}_{\overline{A}} = \Gamma^{(\mathrm{Vo})}_{\overline{A}} = \Gamma^{(\mathrm{diag})}_{\overline{A}} = 0$ and $\Gamma^{\mathrm{S}}_{\overline{A}} = U\Gamma_{\overline{S}}V^\dagger$. Note that this case no longer relies on any properties of S, such that it holds for the dSVD (3.1.4), if the deformed singular value properties hold, as we assumed in the setup.

The last special case drastically simplifies the AD formula and removes the common sources of instabilities, namely from F in the presence of (nearly) degenerate singular values and from S^{-1} in the presence of vanishing (small) singular values. We may use it whenever the truncated SVD as an algorithmic step may be relaxed to a dSVD, e.g. if we only care about the correct subspaces spanned by U and V for truncation, and do not rely on the particular bases for these subspaces that make S diagonal.

If this is not possible, other strategies have been employed to stabilize these expressions. First, note that singular values so small as to make the inverse S^{-1} unstable *should* be truncated and would thus not appear in S, but in Y. Secondly, forming F in the presence of (nearly) degenerate singular values is unstable. One common approach to adress this is to realize that F is always used in conjunction with a factor of S. Therefore, we can instead form

$$G_{ij} := \begin{cases} 0 & i = j \\ 1/(S_i + S_j) & i \neq j \end{cases} \quad ; \qquad H_{ij} := \begin{cases} 0 & i = j \\ 1/(S_i - S_j) & i \neq j \end{cases} , \tag{4.17}$$

such that $FS = (H - G)/2$ and $SF = (H + G)/2$. Forming G is numerically stable, and forming H should be more stable in practice than F. Therefore we can compute the following parts of $\Gamma_{\bar{A}}^{(\mathrm{Uo})}$ and $\Gamma_{\bar{A}}^{(\mathrm{Vo})}$ as

$$(J + J^\dagger)S = \frac{H - G}{2} \circ (U^\dagger \Gamma_{\bar{U}} - \text{h.c.}) \tag{4.18}$$

$$S(K + K^\dagger) = \frac{H + G}{2} \circ (V^\dagger \Gamma_{\bar{V}} - \text{h.c.}) \ . \tag{4.19}$$

Additionally, a broadened inverse $x/(x^2 + \varepsilon)$ is commonly used in place of $1/x$.

For practical implementations, note that the (truncated) SVD is often implemented as a mapping $A \mapsto (U, S, V^\dagger)$, where the third output is $W := V^\dagger$ instead of V. The AD formula for such a function is readily obtained from the result above by substituting $V = W^\dagger$ and $\Gamma_{\bar{V}} = \Gamma_{\bar{W}}^\dagger$ in (4.12).

4.3 Explicit gradient evaluation

Note that the typical cost functions, such as the variational energy (4.1), require approximations to be evaluated, e.g. using the bMPS method discussed in section 2.4.2. Therefore, using automatic differentiation (AD) will yield the gradient *of the approximation*, which is not necessarily a good approximation of the true gradient.

This does not seem to be a problem when optimizing infinite PEPS with the variational energy evaluated using the CTMRG approximate contraction method. The resulting optimization scheme seems to give good results, e.g. in Refs. [38]. For finite PEPS, however, the resulting optimization trajectories when using AD to evaluate the gradients seem unstable.

We propose the following explanation for the instabilities. There is – empirically verified – a substantial region of parameter space, where approximate contraction methods, such as e.g. bMPS contraction, give a good approximation of the true energy of the PEPS at those parameters. However, parameters can be fine-tuned to make the approximations unsound. Now, using AD for the gradients results in a minimization of the quantity that is computed by bMPS contraction and may converge to one of these points outside the region of sound approximation. We have observed this behavior in numerical experiments. For a finite system, using bMPS contraction for the energy and unmodified AD for the gradients, a Limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) optimizer converges to a PEPS whose energy, as computed by the same bMPS scheme, is well below the spectrum of the Hamiltonian, i.e. unphysical. Increasing the bMPS bond dimension beyond its value during optimization reveals this and gives a physical energy, much larger than DMRG reference values for the ground state energy. Thus, we have optimized not for a good ground state but only for parameters that "cheat" the approximate contraction.

Approaches to remedy this behavior have been employed in Ref. [128] but are only heuristically motivated, and it is unclear if they are sufficient in general. Let us emphasize that in light of the approximations in the cost function, the quality of the result as a variational trial state must be judged at a higher accuracy in the approximation, e.g. a higher bMPS bond dimension, than what was used during the optimization. It seems that PEPS optimization obeys Goodhart's law: "When a measure becomes a target, it ceases to be a good measure"[158].

We propose an alternative approach to stabilize the optimization. Instead of differentiating the approximate cost function, we develop approximate contraction methods for evaluating the derivative. To illustrate the difference, let $\mathcal{A}[-]$ denote the approximate contraction of a quantity that involves a tensor network contraction. For example, $\mathcal{A}[E]$ is the result of evaluating the variational energy using approximate contraction. From the AD approach, we would then obtain $\nabla\mathcal{A}[E]$ as the gradient and thus effectively optimize $\mathcal{A}[E]$, which may have its minimum outside the region where $\mathcal{A}[E] \approx E$ is a good approximation. We instead propose to use $\mathcal{A}[\nabla E]$, i.e. writing down an expression for the components of the exact gradient ∇E as a tensor network and then introducing approximations to evaluate them.

The gradient of the variational energy (4.1) is given by

$$\frac{\partial E(|\phi\rangle)}{\partial \overline{A}^{[x,y]}} = \frac{1}{\langle\phi|\phi\rangle}\frac{\partial}{\partial \overline{A}^{[x,y]}}\langle\phi|H|\phi\rangle - E(|\phi\rangle)\frac{1}{\langle\phi|\phi\rangle}\frac{\partial}{\partial \overline{A}^{[x,y]}}\langle\phi|\phi\rangle \tag{4.20}$$

and similarly for the square distance (4.4) we get

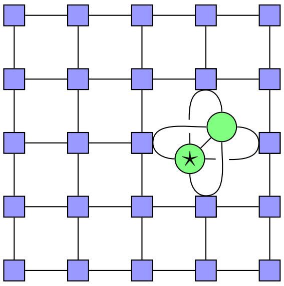
$$\frac{\partial \Delta^2}{\partial \overline{A}^{[x,y]}} = -\frac{1}{\sqrt{\langle\phi|\phi\rangle\langle\psi(t)|\psi(t)\rangle}}\frac{\partial}{\partial \overline{A}^{[x,y]}}\langle\phi|U(dt)|\psi(t)\rangle + \frac{2-\Delta^2}{2}\frac{1}{\langle\phi|\phi\rangle}\frac{\partial}{\partial \overline{A}^{[x,y]}}\langle\phi|\phi\rangle\,. \tag{4.21}$$

In particular, in both cases, we need to evaluate derivatives of the norm, expectation value or of matrix elements. All of those objects depend only linearly on the conjugate tensor $\overline{A}^{[x,y]}$ and thus their derivatives are obtained simply by leaving that tensor out in the bra layer, e.g.

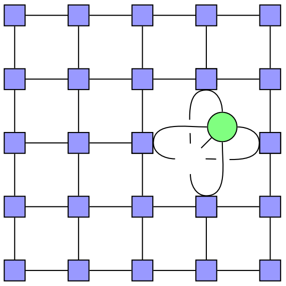
$$\begin{aligned}\frac{\partial}{\partial \overline{A}^{[x,y]}} \langle\phi|\phi\rangle &= \frac{\partial}{\partial \overline{A}^{[x,y]}} \operatorname{TTr}\left[\overline{A}^{[0,0]} A^{[0,0]} \ldots \overline{A}^{[x,y-1]} A^{[x,y-1]} \overline{A}^{[x,y]} A^{[x,y]} \ldots\right] \\ &= \operatorname{TTr}\left[\overline{A}^{[0,0]} A^{[0,0]} \ldots \overline{A}^{[x,y-1]} A^{[x,y-1]} \qquad A^{[x,y]} \ldots\right]\end{aligned} \tag{4.22}$$

for the square norm. This results in a tensor network of the form

$$\frac{\partial}{\partial \overline{A}^{[x,y]}} \langle\phi|\phi\rangle = \frac{\partial}{\partial \overline{A}^{[x,y]}}$$



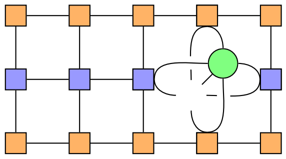
$$=$$



$$, \tag{4.23}$$

where the diagram consists of double layer tensors $F^{[x,y]}$ as defined in (2.55), and we expand the double layer structure only at site (x, y). We now evaluate this diagram using the same bMPS method (2.58) as for the value of the norm. Note that we need to sandwich the row on which the derivative acts with bMPS to find

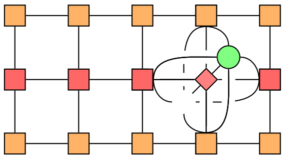
$$\frac{\partial}{\partial \overline{A}^{[x,y]}} \langle\phi|\phi\rangle \approx$$



$$, \tag{4.24}$$

where the top and bottom row are bMPS, approximating the rest of the diagram. For expectation values of a PEPO H with tensors $W^{[x,y]}$ drawn as red diamonds, we find

$$\frac{\partial}{\partial \overline{A}^{[x,y]}} \langle\phi|H|\phi\rangle \approx$$



$$, \tag{4.25}$$

where the red square tensors in the middle row are three-layer tensors (2.57) containing the PEPO. Derivatives of matrix elements $\langle\phi|U(dt)|\psi(t)\rangle$ are analogous.

In this scheme, the bMPS, and partial contractions of e.g. (4.24), can be re-used between the components of the gradient, i.e. between the derivatives $\partial/\partial\overline{A}^{[x,y]}$ w.r.t. tensors on different sites (x, y). Note also that in a gradient-based algorithm, we have an outer loop of the optimization algorithm that suggests a converging sequence of trial parameters. Thus, we may store the bMPS that we find during the evaluation of the cost function or its gradient to use as an initial guess for the variational bMPS method in the next iteration of the outer loop.

If the bMPS environments are used for local expectation values, their prefactors cancel between the numerator and denominator of a normalized expectation value. This is *not* the case here, and we need to explicitly keep track of the bMPS norm.

4.4 Benchmark

Let us now perform a benchmark simulation of the gradient-based methods. We consider the quantum transverse field Ising model (TFIM)

$$H = -\sum_{\langle i,j\rangle} \sigma_i^x \sigma_j^x - g \sum_i \sigma_i^z \tag{4.26}$$

on an $L \times L$ square lattice with open boundary conditions, where σ^α denote Pauli operators with eigenvalues ± 1. The model exhibits a phase transition from a σ^x-ordered ferromagnet for $g < g_c$ to a disordered paramagnet for $g > g_c$ at a critical field strength $g_c \approx 3.05$ [159].

4.4.1 Hamiltonian Representations

To employ the gradient-based methods, we need to represent the relevant operators in a compatible way, e.g. as PEPOs. For ground state search, we can write the Hamiltonian (4.26) either as a sum of $\sim L^2$ bond operators, as a sum of $2L$ MPOs, or the sum of 2 PEPOs. In the following, we provide explicit constructions for all of these representations.

Sum of bond operators First, we find

$$H = \sum_{\langle i,j\rangle} h^{[ij]} \qquad h^{[ij]} := -\sigma_i^x \sigma_j^x - \frac{1+\lambda_{\langle i,j\rangle}}{4} g\sigma_i^z - \frac{1+\rho\langle i,j\rangle}{4} g\sigma_j^z \ , \tag{4.27}$$

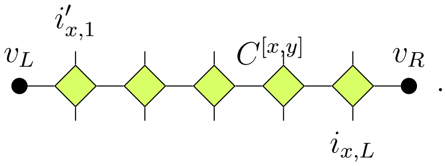
where $\lambda_{\langle i,j\rangle}, \rho_{\langle i,j\rangle} \in \{0,1\}$ correct for double counting of sites by setting $\lambda_{\langle i,j\rangle} = 1$ iff site i is at an open boundary and similarly $\rho_{\langle i,j\rangle} = 1$ iff j is at a boundary.

Sum of MPOs Alternatively, we may use the finite state machine construction [89] to write all terms within a single row or single column as an MPO, that is

$$H = \sum_{x=1}^{L} H^{[x,:]} + \sum_{y=1}^{L} H^{[:,y]} \ , \tag{4.28}$$

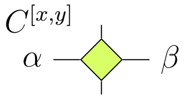
where the operators act on only a single row (column) and are given as MPOs with the following coefficients

$$\left\langle i_{x,1} \dots i_{x,L} \middle| H^{[x,:]} \middle| i'_{x,1} \dots i'_{x,L} \right\rangle \ =$$



(4.29)

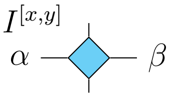
The MPO tensors are given by



$$:= \begin{pmatrix} \mathbb{1} & \sigma^x & -\frac{1}{2} g\sigma^z \\ 0 & 0 & -\sigma^x \\ 0 & 0 & \mathbb{1} \end{pmatrix}_{\alpha\beta} \ , \tag{4.30}$$

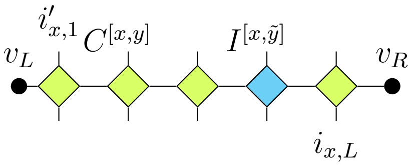
where we suppress the physical indices and identify them with the indices that the operator-valued entries of the matrix have in the computational basis. The boundary vectors are $v_L = (1\ 0\ 0)^{\mathrm{T}}$ and $v_R = (0\ 0\ 1)^{\mathrm{T}}$.

Sum of PEPOs Lastly, we can employ a finite state machine MPO construction to form a PEPO that is the sum of all horizontal MPOs above. Note that if we replace a single MPO tensor, e.g. at position $\tilde{y}$ with



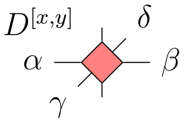
$$:= \begin{pmatrix} 0 & 0 & \mathbb{1} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}_{\alpha\beta} , \tag{4.31}$$

the modified MPO



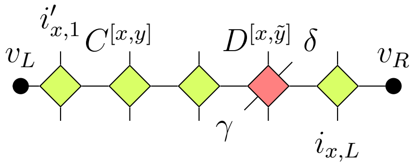
$$= \left\langle i_{x,1} \ldots i_{x,L} \middle| \mathbb{1}^{[x,:]} \middle| i'_{x,1} \ldots i'_{x,L} \right\rangle \tag{4.32}$$

gives us the identity operator on the same row. Now, define the six-leg tensor



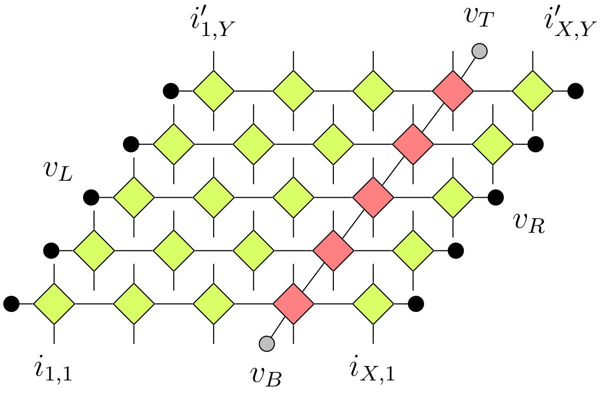
$$:= \begin{pmatrix} I^{[x,y]}_{\alpha\beta} & C^{[x,y]}_{\alpha\beta} \\ 0 & I^{[x,y]}_{\alpha\beta} \end{pmatrix}_{\gamma\delta} . \tag{4.33}$$

We find the following intermediate result, where we substitute one MPO tensor at some arbitrary posititition $1 < \tilde{y} < L$ for a D tensor to get the MPO-like tensor network with an extra pair of virtual legs



$$= \left\langle i_{x,1} \ldots i_{x,L} \middle| \begin{pmatrix} \mathbb{1}^{[:,y]} & H^{[:,y]} \\ 0 & \mathbb{1}^{[:,y]} \end{pmatrix}_{\gamma\delta} \middle| i'_{x,1} \ldots i'_{x,L} \right\rangle . \tag{4.34}$$

Thus, with the boundary vectors $v_B = (1\ 0)^{\mathrm{T}}$ and $v_T = (0\ 1)^{\mathrm{T}}$, we can construct the PEPO



$$= \left\langle \{i_{x,y}\} \middle| H_{\mathrm{hor}} \middle| \{i'_{x,y}\} \right\rangle \tag{4.35}$$

for the horizontal terms in

$$H = H_{\mathrm{hor}} + H_{\mathrm{vert}} := \sum_{y=1}^{L} H^{[:,y]} + \sum_{x=1}^{L} H^{[x,:]}. \tag{4.36}$$

An analogous construction applies to H_{vert}.

Choosing the representation comes down to a trade-off between the number of terms that make up the whole Hamiltonian and how difficult it is to deal with each such term. For bMPS contraction, using MPOs is not significantly harder than using bond operators and should thus always be preferred. For evaluating the expectation value, using the MPOs is convenient since then the bMPS from the two-layer norm diagram may be used to sandwich the column or row on which the MPOs acts. This is not possible, however, for evaluating gradients, and we have found using PEPOs most practical in that case.

4.4.2 Evolution Operator Representations

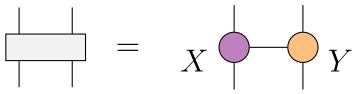
For the time evolution method, we require an expression for the time evolution operator $U = \mathrm{e}^{-\mathrm{i}H\delta t}$ as a PEPO, usually in a Trotter(-like) approximation for small δt. We explicitly provide constructions for two different approaches.

From product of bond gates First, consider a Trotterization of the sum of bond operators (4.27). We can identify four groups of bonds, such that within each group, bonds do not overlap, and thus, the bond operators $h^{[ij]}$ commute. We choose as the first two groups the horizontal bonds with even (odd) x coordinate of the left site, and as the remaining two groups the vertical bonds with even (odd) y coordinate of the bottom site. We write α to denote one such group of bonds. We find

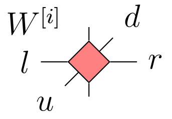
$$\mathrm{e}^{-\mathrm{i}H\delta t} = \exp\left[-\mathrm{i}\sum_{\alpha}\sum_{\langle i,j\rangle\in\alpha} h^{[ij]}\delta t\right] \approx \prod_{\alpha}\exp\left[-\mathrm{i}\sum_{\langle i,j\rangle\in\alpha} h^{[ij]}\delta t\right] = \prod_{\alpha}\prod_{\langle i,j\rangle\in\alpha}\mathrm{e}^{-\mathrm{i}h^{[ij]}\delta t}, \tag{4.37}$$

where the approximation is up to corrections in $\mathrm{O}(\delta t^2)$. Now, to embed this into a PEPO, we require a factorization

$$\mathrm{e}^{-\mathrm{i}h^{[ij]}\delta t} =: \sum_{k=1}^{\eta} X_k^{[i],j} Y_k^{[j],i} \quad ; \quad \mathrm{e}^{-\mathrm{i}h^{[ij]}\delta t} \tag{4.38}$$

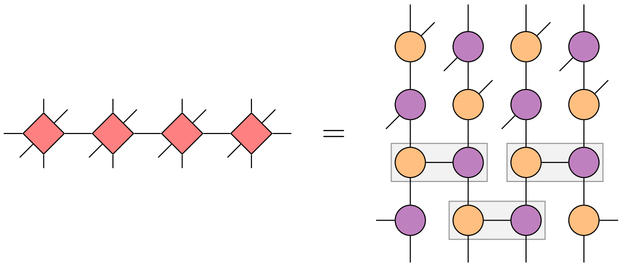


of the bond gate in terms of two three-leg tensors. Here, the superscript in brackets indicates which site is acted on, while the second superscript identifies the second site of the bond that $h^{[ij]}$ acts on. Such a factorization is always possible at rank $\eta \leq d^2 = 4$ with (truncated) factorizations. For the TFIM at finite filed strength $g > 0$, we require $\eta = 4$. We can now define PEPO tensors



$$:= \begin{cases} X_u^{[i],i+e_y} Y_u^{[i],i-e_y} X_u^{[i],i+e_x} Y_u^{[i],i-e_x} & i \in A \\ Y_u^{[i],i-e_y} X_u^{[i],i+e_y} Y_u^{[i],i-e_x} X_u^{[i],i+e_x} & i \in B \end{cases}, \tag{4.39}$$

where the difference between the cases for A and B sublattices is only a swapped order of the operators. Note that we again suppress the physical indices associated with the vertical legs and associate them with the operators on the RHS. As a result, a PEPO (2.54) built from these tensors realizes (4.37). To see this, consider the following patch of the larger PEPO network

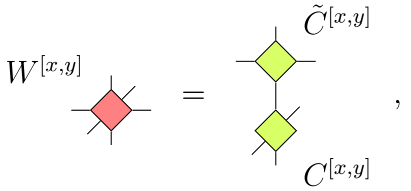


(4.40)

and observe that the bond operators indicated by gray background boxes each multiply to the gate $\mathrm{e}^{-\mathrm{i}h^{[ij]}\delta t}$ on the respective bond.

As a result, we have constructed a PEPO for the evolution operator of the TFIM with bond dimension $\eta = 4$.

From product of MPOs Alternatively, we may first Trotterize between the group of horizontal and the group of vertical MPOs in equation (4.28). Within these groups, the MPOs mutually commute, and we can take the exponential of each MPO according to the $W^{I/II}$ methods of Ref. [102]. The resulting MPO has a bond dimension one smaller than for the corresponding Hamiltonian, i.e. $\eta = 2$ for the TFIM. Assume that $C^{[x,y]}$ for $y = 1, \dots, L$ are the MPO tensors for the approximate exponential $\mathrm{e}^{-\mathrm{i}H^{[x,:]}\delta t}$ of the vertical MPO on column x, and similarly $\tilde{C}^{[x,y]}$ the tensors for $\mathrm{e}^{-\mathrm{i}H^{[:,y]}\delta t}$ on row y. We define a PEPO tensor as



(4.41)

which is designed such that on forming the PEPO tensor network, the MPO tensors connect in such a way as to form the exponentials of row (column) Hamiltonians. In particular, they parameterize a Trotterized time evolution PEPO

$$\begin{aligned}
\mathrm{e}^{-\mathrm{i}H\delta t} &\approx \mathrm{e}^{-\mathrm{i}H_{\mathrm{hor}}\delta t}\mathrm{e}^{-\mathrm{i}H_{\mathrm{vert}}\delta t} = \left(\prod_{y=1}^{L} \mathrm{e}^{-\mathrm{i}H^{[:,y]}\delta t}\right)\left(\prod_{x=1}^{L} \mathrm{e}^{-\mathrm{i}H^{[x,:]}\delta t}\right) \\
&\approx \left(\prod_{y=1}^{L} \mathrm{MPO}(\tilde{C}^{[1,y]} \dots \tilde{C}^{[L,y]})\right)\left(\prod_{x=1}^{L} \mathrm{MPO}(C^{[x,1]} \dots C^{[x,L]})\right) \\
&= \mathrm{PEPO}(\{W^{[x,y]}\})
\end{aligned} \tag{4.42}$$

with bond dimension $\eta = 2$. Again, the approximation is up to corrections in $\mathrm{O}(\delta t^2)$.

4.4.3 Ground State Results

As a benchmark of the gradient-based ground state search, we optimize ground state PEPS for the TFIM (4.26) at a field strength $g = 3$, which is close to criticality. For an initial guess with bond dimension $D = 2$, we apply a $D = 2$ PEPO (4.42) of a small step $\mathrm{i}\delta t = \delta\tau \sim 0.01$ of *imaginary* time evolution to a $\sigma^z = +1$ product state. We then optimize at this bond dimension by minimizing the variational energy (4.1) until convergence. For optimizations at larger PEPS bond dimension, we start from the $D = 2$ result and increase its bond dimension, either by padding with zeroes or by applying another imaginary time step exactly. In either case, we found it helpful for stability to introduce a small random perturbation and to apply a random gauge transformation on the bonds to avoid vanishing tensor entries. Without this step, i.e. just embedding a $D = 2$ PEPS into $D = 3$, by padding with zeroes, all derivative components corresponding to these new slices will vanish, and we will be stuck at a saddle point for these entries.

L	D	This Work	FU	isoTNS	DMRG
11	2	−3.17185	−3.17128	−3.15546	−3.17211
	3	−3.17185	−3.17210		
	4	−3.17206	−3.17210	−3.16625	
20	2	−3.18138			−3.18197
	3	−3.18138			
	4	−3.18147			
21	2	−3.18193	−3.18128		
	3	−3.18193	−3.18242		
	4	−3.18201	−3.18243		

Table 4.1: Variational ground state energy per site E/L^2 of the TFIM at $g = 3$ on an $L \times L$ square lattice with open boundary conditions. For this work and the Full Update (FU) algorithm [107], the variational manifold are PEPS with bond dimension D. The DMRG^2 algorithm [71] (denoted isoTNS) finds a ground state approximation within the manifold of isometric PEPS of bond dimension D, a strict subset of all PEPS. For comparison, we also give energies obtained from MPS calculation, using the DMRG algorithm for which D is meaningless. For the 11×11 system, we used TeNPy [3, 4], while the 20×20 data was obtained in Ref. [32] using specialized hardware (TPUs) and massive computational power at MPS bond dimension $2^{16} = 65536$.

In table 4.1, we compare variational energies as a measure of quality to other TNS methods. At the lowest non-trivial bond dimension of $D = 2$, we have achieved better ground state energies than any other PEPS work we are aware of. In particular, we improve upon the FU results of Ref. [107] significantly and by more than the margin of error of the approximately evaluated energy. We have fully converged the result at $L = 11$, using a bMPS bond dimension $\chi_{\max} = 300$ for the last few optimization steps close to the minimum, and evaluated the energies at $\chi_{\max} = 350$. We can conjecture with reasonable certainty that the result is, up to numerical precision, the global optimum within the manifold of PEPS with bond dimension $D = 2$. For the larger systems, our proof-of-principle implementation reaches the limits of its computational performance. Reaching a gradient norm of ~ 0.1 at $\chi_{\max} = 200$, we are close to, but not fully converged, and still see change in the energy density on the fifth digit. At the larger bond dimensions $D = 3, 4$, the results are not fully converged. At $D = 4$, we were only able to perform ~ 100 optimizer steps for $L = 11$, and only ~ 10 steps for $L = 20, 21$ after applying an imaginary time step to the $D = 2$ result. We still obtain a competitive quality result.

4.4.4 Time Evolution Results

As a benchmark for the gradient-based time evolution algorithm, we simulate the time evolution induced by the TFIM Hamiltonian (4.26) after a local quench. For the quench, we apply $\sigma^y_{\boldsymbol{c}}$ to the central site $\boldsymbol{c}$ of a 11×11 lattice. This constitutes a spin-flip excitation in both limits of the phase diagram. We then simulate the dynamics

$$|\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}Ht}\sigma^y_{\boldsymbol{c}} \, |\mathrm{GS}\rangle \tag{4.43}$$

by applying a sequence of time steps $U(\delta t)$, giving us access to the evolved state at a grid of discrete times $t_n = n\delta t$. We then extract the time-dependent correlation function

$$C^{yy}(\boldsymbol{r}, t) := \langle \mathrm{GS}|\sigma^y_{\boldsymbol{r}}(t)\sigma^y_{\boldsymbol{c}}|\mathrm{GS}\rangle \, . \tag{4.44}$$

We evaluate it from the quench dynamics as

$$C^{yy}(\boldsymbol{r}, t_n + \tau) = \mathrm{e}^{\mathrm{i}E_0\tau} \left\langle \mathrm{GS}(t_n) \middle| \sigma^y_{\boldsymbol{r}}\mathrm{e}^{-\mathrm{i}H\tau} \middle| \psi(t_n) \right\rangle . \tag{4.45}$$

Here, we may explicitly include a time evolution operator for some smaller time step $|\tau| \leq \delta t/2$ to increase the resolution at which we evaluate the correlation function. This is does not dominate the cost, as (4.45) is a tensor network with the same structure as the cost function for time evolution. It also allows for a consistency check of the approximate time evolution and Trotter approximation, where a kink in the data from $t_n + \tau_{\max}$ to $t_{n+1} - \tau_{\max}$ reveals problems in one of the two approximations. Note that bMPS and partial contractions can be re-used between the evaluation for different positions $\boldsymbol{r}$. We observe that it is beneficial to use the same time evolution method to explicitly evolve the ground state approximation instead of assuming that $\langle \mathrm{GS}|\mathrm{e}^{\mathrm{i}Ht} = \mathrm{e}^{\mathrm{i}E_0t}\langle \mathrm{GS}|$. This is because (a) the ground state is only an approximation, and thus only approximately an energy eigenstate, and (b) the time evolution contains approximations and does not conserve energy exactly.

We then form the dynamical spin structure factor (DSF)

$$S^{yy}(\boldsymbol{k}, \omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{d}t \, \sum_{\boldsymbol{r}} \mathrm{e}^{\mathrm{i}(\omega t - \boldsymbol{k}\cdot\boldsymbol{r})} C^{yy}(\boldsymbol{r}, t). \tag{4.46}$$

Let us first elaborate on the modifications needed to compute a proxy for the DSF from the available data. Firstly, to get access to correlation data for negative times, we may observe that in an infinite system with translational invariance, the correlation function fulfills

$$\begin{aligned}C^{yy}(\boldsymbol{r},-t) &= \langle \mathrm{GS}|\sigma^y_{\boldsymbol{r}}(0)\sigma^y_{\boldsymbol{c}}(t)|\mathrm{GS}\rangle = \overline{\langle \mathrm{GS}|\sigma^y_{\boldsymbol{c}}(t)\sigma^y_{\boldsymbol{r}}(0)|\mathrm{GS}\rangle} = \overline{C}^{yy}(-\boldsymbol{r},t)\\ &= \overline{C}^{yy}(\boldsymbol{r},t),\end{aligned} \tag{4.47}$$

where we used invariance of the ground state expectation value under time shifts, hermicity of the σ^y, translational invariance, and inversion symmetry. As a result, we may evaluate the DSF as

$$S^{yy}(\boldsymbol{k},\omega) = \sum_{\boldsymbol{r}} \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{r}} \frac{1}{\pi} \int_0^\infty \mathrm{d}t \ \mathrm{Re}\left[\mathrm{e}^{\mathrm{i}\omega t} C^{yy}(\boldsymbol{r},t)\right] \ . \tag{4.48}$$

While this identity only holds for the infinite system, and translational invariance is not given for the finite system, we may choose to extrapolate the expression (4.48) to the thermodynamic limit instead of (4.46). Alternatively, one could repeat the simulation to negative times by applying a sequence of $U(-\delta t)$ operators.

Secondly, we only have correlation data up to some finite cutoff time T, that is only for $t \leq T$. Thus, for the time-to-frequency Fourier transform, e.g. for a general function f defined as

$$\hat{f}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \mathrm{d}t \ \mathrm{e}^{\mathrm{i}\omega t} f(t) \tag{4.49}$$

we need to introduce some approximations to deal with the $|t| > T$ integration range. If we simply truncate the integration range, effectively including a windowing function $W(t) = \theta(T - |t|)$, we would find

$$\frac{1}{2\pi} \int_{-T}^{T} \mathrm{d}t \ \mathrm{e}^{\mathrm{i}\omega t} f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \mathrm{d}t \ \mathrm{e}^{\mathrm{i}\omega t} f(t) W(t) = (\hat{f} * \hat{W})(\omega) \ , \tag{4.50}$$

where $*$ denotes the convolution

$$(\hat{f} * \hat{W})(\omega) = \int_{-\infty}^{\infty} \mathrm{d}y \ \hat{f}(y)\hat{W}(\omega - y) \tag{4.51}$$

of two functions. We obtain the desired result $\hat{f}(\omega)$, but convoluted, i.e. broadened by a sinc function $\hat{W}(\omega) = 2/\omega \sin \omega T$. This function has oscillating and slowly decaying tails around a main peak, such that sharp features in $\hat{f}(\omega)$ would be surrounded by ringing artifacts, which makes it difficult to interpret features. We employ a common approach to avoid these by choosing a different windowing function, in particular, a Gaussian $W_\sigma(t) = \mathrm{e}^{-t^2/2\sigma^2}$ which results in convolution with

$$\hat{W_\sigma}(\omega) = \sqrt{\frac{\sigma}{2\pi}}\, \mathrm{e}^{-\sigma^2\omega^2/2}, \tag{4.52}$$

i.e. a Gaussian with width $1/\sigma$. This is a significant improvement since $\hat{W_\sigma}$ decays significantly faster, is strictly decreasing for $\omega > 0$, and is positive, such that convoluting with it does not introduce ringing and only smooths out the features. We

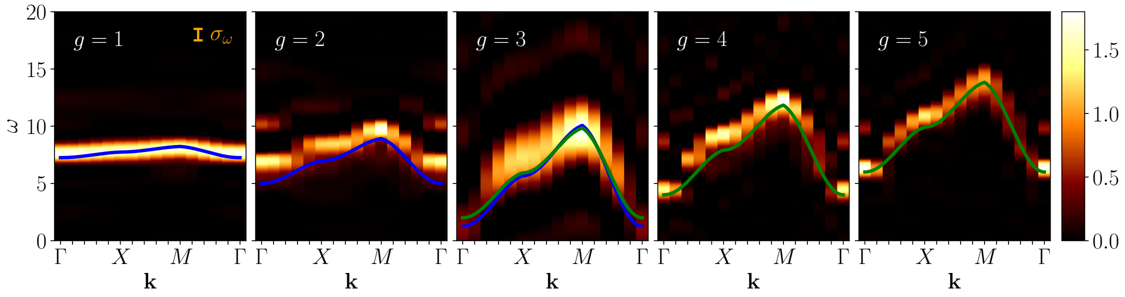


Figure 4.1: The dynamical structure factor $S^{yy}(\mathbf{k},\omega)$ of the TFIM at different field strengths on an 11×11 square lattice with open boundary conditions. The horizontal axis describes a closed triangular path through the Brillouin zone with corners $\Gamma = (0,0)$, $X = (\frac{10}{11}\pi, 0)$, $M = (\frac{10}{11}\pi, \frac{10}{11}\pi)$. The lines are the perturbative dispersions for low (blue) and high (green) field strengths, i.e. equations (4.55) and (4.56). The width σ_ω of the Gaussian broadening that we artificially introduce is indicated in the left panel.

choose the width such that $W_\sigma(T) \ll 1$ is sufficiently decayed, such that we may truncate the integration range to $|t| < T$ and introduce only a small error.

Lastly, we only have access to correlation data at discrete times, such that we need to discretize the integral. Including these three modifications, we can evaluate the broadened structure factor as

$$\hat{W_\sigma}(\omega) * S(\boldsymbol{k},\omega) \approx \sum_{\boldsymbol{r}} \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{x}} \operatorname{Re}\left[\frac{1}{N}\sum_{n=0}^{N} \mathrm{e}^{\mathrm{i}\omega t_n} C^{yy}(\boldsymbol{r},t_n)\right]. \tag{4.53}$$

We present results for the DSF from gradient-based time evolution in figure 4.1. The extracted structure factor matches well with the perturbative dispersion relations. At $g = 3$, we see a small but significant gap of $\Delta \approx 1$ at the Γ point. We do not expect to see the gap close exactly because we simulate (i) slightly off criticality $g = 3 \lesssim g_c$, (ii) at finite system size, and (iii) at finite entanglement (due to the bounded bond dimension). The results agree well with other works, such as isometric TNS in reference [71] and infinite PEPS in reference [133], which perform very similar simulations of the DSF, as well as qualitatively with the infinite projected entangled pair state (iPEPS) excitation ansatz [160, 161], which extracts the excitation spectrum (i.e. the dispersion relation) in the thermodynamic limit. We attribute the faint features above the main branch at $g = 1, 2$ to two-magnon bound states; the energy scale agrees with the result from the excitation ansatz [161].

We can obtain a reference for the DSF from perturbative calculations. In the limit of small transverse fields, $g \ll g_c$, the elementary excitations on top of the σ^x polarized ground state are single spin flips, and the simulated quench does indeed create such an excitation. Now, second-order time-dependent perturbation theory in the field term results in a hopping model of these spin flips, where nearest neighbor hopping is favored since the intermediate virtual state of two neighboring spin flips dissatisfies the interaction term on only six bonds, compared to the eight bonds for other configurations. In the opposite limit $g \gg g_c$, the quench also induces a spin-flip, this time on top of a $\sigma^x = +1$ polarized ground state, and we directly find a nearest neighbor hopping model of the spin flips in first-order perturbation theory of the interaction term. In both cases, we find sharp features

$$S^{yy}(\boldsymbol{k}, \omega) = \delta(\omega - \epsilon(\boldsymbol{k})) \tag{4.54}$$

in the structure factor with weight on the dispersion relations, which are given by

$$\epsilon_{g \ll g_c}(\mathbf{k}) = 8 - \frac{g^2}{4}\left[1 + \cos k_x + \cos k_y\right] + \mathrm{O}\left(g^3\right) \tag{4.55}$$

$$\epsilon_{g \gg g_c}(\mathbf{k}) = 2g\left(1 - \frac{1}{g}\left[\cos k_x + \cos k_y\right] + \mathrm{O}\left(\frac{1}{g^2}\right)\right). \tag{4.56}$$

These dispersion relations are overlayed in figure 4.1 for comparison.

4.5 Conclusion

We have proposed an approach to the gradient-based optimization of tensor network states, in particular finite PEPS, in light of approximate contractions. This avoids the pathological states that can result from naively optimizing an approximately contracted cost function using automatic differentiation. By instead evaluating the exact expression of the gradient of the cost function using the same approximate contraction method, we have achieved a stable and successful optimization algorithm. In this proof of principle work, we do not exploit symmetries, limiting the bond dimension achievable on larger systems. Nevertheless, we see better results than any other publication (for *finite* PEPS) we are aware of at the lowest bond dimension $D = 2$, where we reach convergence and achieve comparable energies at the larger bond dimensions. Notably, our ground state results, simulated sequentially on tens of CPU cores, yield results competitive with the massively parallel DMRG simulation of Ref. [32] running on a thousand cores of specialized TPU hardware. This discrepancy further illustrates the potential of natively two-dimensional tensor network methods, such as PEPS, for the simulation of 2D quantum systems. The gradient-based approach allowed us to perform dynamics simulations even at the lowest bond dimensions, which – to our knowledge – do not admit accurate time evolution by other means.

Our results showcase conventional wisdom established when comparing simple update (SU) and full update (FU) ground state searches that the maximal bond dimension reached by a PEPS simulation only loosely correlates with the quality of the results. We observe that local updates do not fully exhaust the variational power of the fixed bond dimension manifold. The global, gradient based method on finite system offers an approach complementary to the TEBD-style local updates on large unit cell infinite systems of Ref. [133], to extract spectral functions in 2D. Offering a competitive alternative to MPS simulations on cylinder geometries, especially when considering more strongly correlated models than the TFIM, is an open challenge.

Nevertheless, gradient-based approaches seem to be a promising avenue to leverage the full variational power of the PEPS ansatz, depending on how far performance can be pushed. In any case, hybrid approaches may prove fruitful, leveraging the performance of established algorithms, such as e.g. FU or similar, to get a good approximation, then further improving on it with a few costly, but effective gradient steps, as is done in Ref. [115]. The different convergence properties of global energy minimization and imaginary time evolution may also complement each other to avoid local minima and plateaus.

The future directions are to push the performance to fully converge the ground state results for the larger systems and larger bond dimensions, and we refer future readers to a future published version of [2]. We also plan to study the implications of the simplification of the AD formula of the truncated SVD, given in section 4.2.2 for gradient-based optimization using gradients from AD, and if it might stabilize the optimization. It may also prove beneficial for performance to employ the GPU-friendly truncation routines discussed in chapter 3 in the bMPS contraction to exploit hardware acceleration.

Chapter 5

Non-abelian symmetries and anyons in tensor networks

In this chapter, we develop a mathematical framework that allows non-abelian symmetries to be exploited in tensor network simulations. We choose a quite general approach using category theory. This is in contrast to approaches [52, 115, 162] that focus on the particular properties of SU(2), the most common non-abelian symmetry in condensed matter systems. As a result, the framework for symmetric tensors extends seamlessly from expressing quantum states that are symmetric under some symmetry group to states with fermionic or anyonic statistics. Enforcing a group symmetry results in computational and memory benefits as a result of Schur's lemma. For group symmetries, there is a notion of "general" tensors in an ambient space that has no symmetry constraints, and enforcing the symmetry restricts the entries of this tensor, namely by the charge rule (2.70), and additionally by restricting components between symmetry sectors to the identity, which only has non-trivial consequences in the non-abelian case. In the general case, and in particular for fermions or anyons, there is no notion of such an ambient space, and symmetric tensors are the only tensors we can write down. We can think of the generalized Schur's lemma as a way to construct or parametrize the symmetric tensors that makes operating on them convenient. Throughout this chapter, we slightly abuse the term symmetry by generalizing it to the fermionic or anyonic case. We understand "symmetry" here to mean the mathematical structure that constrains the form of allowed (meaning symmetric) tensors, which is either the symmetry group or the tensor category that the *tensors* live in. It is not to be confused with the notion of a categorical symmetry [163], where the symmetry transformations themselves live in a particular category – a related but distinct notion.

The mathematical foundation for this general framework is monoidal category theory, which may be prohibitively involved to learn solely for the purpose of understanding, e.g. SU(2) symmetric tensors. Therefore, we attempt to offer two complementary ways of reading this chapter. On the one hand, we aim for an intuitive approach that focuses solely on the case of a symmetry group. This follows the spirit of Steven Simon's approach of "avoid[ing] the language of category theory like the plague" [164]. On the other hand, we aim to provide a sufficiently rigorous approach to make all concepts unambiguously well-defined from a mathematical perspective. For the most part, we attempt to balance these approaches so that statements make sense within the limited scope of the first approach while still being correct in the general case. Wherever that is impractical, we split the text into side-by-side columns and give concrete explanations or definitions separately from the two separate perspectives. The text in the left columns avoids category theory and defines the concepts purely in terms of group representations, while the right columns introduce and use monoidal category theory.

In section 5.1, we introduce the basic definitions regarding symmetric maps. We streamline the exposition to the concepts and structures needed for the purpose of a tensor backend and establish a graphical language that allows the intuition from the concrete case of a group symmetry to carry over to the general categorical case. We define and identify the pieces of data that are required of a symmetry to be used in this framework – its *topological data* – in section 5.2. In section 5.3, we identify the free parameters of symmetric tensors, propose a storage format, and develop in detail how to perform common operations on these tensors. We remark on implementation details and upcoming plans to integrate the framework into the TeNPy library in section 5.4 before showing benchmark results in section 5.5 and concluding in section 5.6.

The developed framework is informed by the implementation and documentation of TensorKit [165], a Julia library for symmetric tensors. The exposition of category theory is largely based on Ref. [166], which we would like to recommend as literature for an approach to category theory from a quantum information perspective, as well as [164, 167]. We would also like to recommend Ref. [168] for a detailed review of the graphical notation for monoidal categories, as well as point to introductions to category theory from a perspective of topological excitations in Refs. [169, 170].

5.1 Definitions and graphical language

In the following section, we aim to introduce the basic concepts of symmetric tensors in both an accessible and a general way. Whenever these approaches are incompatible, we split into columns, focusing on the concrete case where we assume that the symmetry is given by a *group*, acting on the Hilbert space via a representation in the right columns and a general case where it is encoded in a *category* in the left column. The former allows us to rely on only a minimal background of linear algebra and to give concrete, constructive definitions or at least examples. The latter is generally phrased axiomatically and allows us to be very general.

Concrete case: Group Representation

In the concrete case, we assume that the context implies a symmetry, which is given by a group G, the *symmetry group*. We assume that G is either finite or a compact

General case: Tensor Category

In the general case, the "symmetry" is encoded in a category $\mathbf{C}$.

The full list of properties required of the

Lie group. This is the case for the symmetries that commonly arise in condensed matter systems.

We assume that we are working with complex, finite-dimensional Hilbert spaces. Assuming an algebraically closed field is required for Schurs's lemma part 2 to apply. Assuming a finite dimension allows us to work with finite sums instead of infinite series, where e.g. convergence and commutation of sums needs to be checked. This is natural in tensor networks, where all bond dimensions are finite. Finally, assuming the existence of an inner product $\langle - | - \rangle$ is natural in quantum mechanics and simplifies the construction of matrix representations, etc.

category to be compatible with the framework of symmetric tensors is quite the jargon soup; We require **C** to be a braided pivotal spherical rigid semisimple $\mathbb{C}$-linear monoidal dagger category. We refer to such categories as *tensor categories*. Note that the term is loaded and understood to mean slightly different things in different contexts. We introduce the defining structures of a tensor category in the following sections. See section 5.1.10 for an overview.

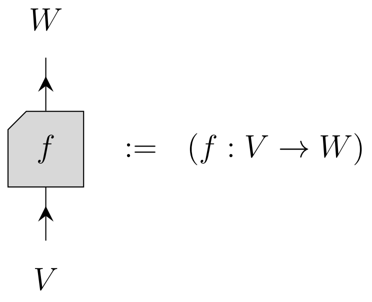
For concrete examples, we may think of the category $\mathbf{FdVect}_{\mathbb{C}}$ of finite-dimensional vector spaces over the complex numbers, which models the trivial symmetry or "no symmetry". A symmetry group (the concrete case discussed in the left columns) is modeled by the category $\mathbf{FdRep}_{\mathbb{C}}(G)$ of finite-dimensional representations of the symmetry group G over the complex numbers. As a notable non-group example, consider the category **Ferm**, which results from equipping the category $\mathbf{FdSVect}_{\mathbb{C}}$ of finite-dimensional complex super vector spaces with a non-trivial twist. As the name suggests, it models fermionic degrees of freedom.
Lastly, consider the category **Fib**, describing Fibonacci anyons. See appendix A for details.

The following subsections each introduce a (group of related) concept(s). In the full-width main text, we summarize its purpose and the intuition behind it, as well as its graphical representation and state relevant properties. We give concrete definitions in the two respective columns.

5.1.1 Spaces and Maps

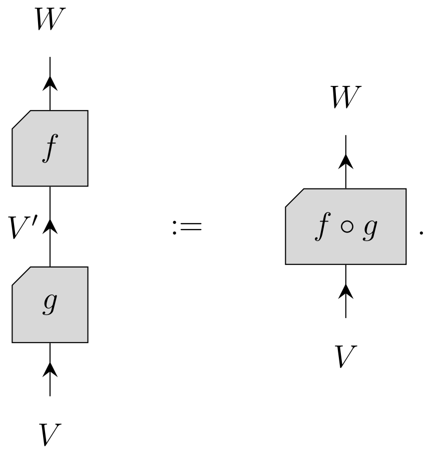
As a first building block for tensors in a tensor network, we consider the physical or virtual spaces that define the legs of a tensor. We understand them as structured sets, e.g. having the structure of a vector space with a grading into sectors, induced by the symmetry. We call these structured sets *symmetry spaces*, even if, in the general anyonic case, they are not vector spaces in the usual sense. Secondly, we need the concept of symmetry-preserving maps, which we call *symmetric maps* for short. They are maps $f : A \to B$ between symmetry spaces A and B. At this point, we can think of the maps as matrices, i.e. two-leg tensors. For multi-leg tensors, we require the tensor product structure to be introduced in the next section.

There is a graphical calculus for maps. Symmetry spaces are represented by wires with arrows. In a later section on duality, we introduce downward arrows as well. Normal arrows point upward in the reading direction. A map $f : V \to W$ is drawn as a box with the domain V as a wire going into the bottom and the codomain coming out from the top.



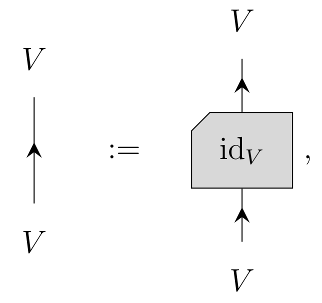
(5.1)

The box for the map f has a chamfered top left corner. This allows us to visually distinguish mirroring and rotation, which we introduce later. Map composition is drawn as vertical stacking; that is for $f : V' \to W$ and $g : V \to V'$, the composite is drawn as



(5.2)

We draw the identity map as an empty wire



(5.3)

which makes its defining property (5.8) as the unit of composition visually apparent.

Concrete case: Group Representation

A *symmetry space* in the above sense is a finite-dimensional complex Hilbert space V equipped with a unitary representation

$$U_V : G \to \mathrm{Hom}\,(V,\ V)$$

of the symmetry group G, which assigns to every group element $g \in G$ a unitary linear map $U_V(g) : V \to V$ such that

$$U_V(gh) = U_V(g) \circ U_V(h)$$

for all $g, h \in G$. See section A.1 for a summary of results from group representation theory.

A *symmetric map* in the above sense is a linear map $f : V \to W$ between symmetry spaces V and W that is compatible with the symmetry representations on the respective spaces, meaning

$$f \circ U_V(g) = U_W(g) \circ f \quad \forall g \in G. \tag{5.4}$$

Linear maps f with this property are also known as equivariant, as intertwiners between U_V and U_W, or as G-linear. The identity map $\mathbb{1} : V \to V, v \mapsto v$ has this property and thus is a symmetric map, and if f and g are symmetric maps, so is $f \circ g$.

We call two symmetry spaces V, W *isomorphic* if there is an invertible symmetric map between them. Note that this is a stronger requirement than an isomorphism of vector spaces. The invertible isomorphism $S : V \xrightarrow{\cong} W$, in addition to being linear must also fulfill (5.4), i.e.

$$U_V(g) = S^{-1} \circ U_W(g) \circ S \ \ \forall g \in G. \tag{5.5}$$

This, in turn, means that U_V and U_W are equivalent as group representations.

It is a common pattern to define concrete symmetric maps “by linear extension”. To fully specify a linear map $f : V \to W$, it is enough to specify the images $f(v_i)$ of a complete subset $\{v_i\} \subseteq V$, e.g. a basis. Since the subset is complete, any $v \in V$ can

General case: Tensor Category

In the general case, the notions of maps and spaces arise from the definition of a category. This column summarizes basic definitions of category theory. A category $\mathbf{C}$ consists of the following data;

- A collection $\mathrm{Ob}(\mathbf{C})$ of *objects*. These are the “symmetry spaces”.
- For every pair of objects A, B, a collection $\mathbf{C}(A, B)$ of *morphisms*, which are denoted $f : A \to B$. These are the “symmetric maps”.
- For every pair of morphisms $f : A \to B$ and $g : B \to C$ with common intermediate object, a *composite* morphism $f \circ g : A \to C$.
- For every object A, an *identity* morphism $\mathrm{id}_A : A \to A$.

which fulfills the axioms of associativity

$$h \circ (g \circ f) = (h \circ g) \circ f \tag{5.7}$$

and identity

$$\mathrm{id}_B \circ f = f = f \circ \mathrm{id}_A \tag{5.8}$$

for all objects $A, B, C, D \in \mathrm{Ob}(\mathbf{C})$ and morphisms $f : A \to B$, $g : B \to C$ and $h : C \to D$.

An *isomorphism* $f : A \to B$ is a morphism that is invertible, which means that there is an $f^{-1} : B \to A$ such that $f^{-1} \circ f = \mathrm{id}_A$ and $f \circ f^{-1} = \mathrm{id}_B$. If an isomorphism $A \to B$ exists, we say that $A \cong B$ are *isomorphic*.

Given categories $\mathbf{C}$ and $\mathbf{D}$, a (covariant) *functor* $F : \mathbf{C} \to \mathbf{D}$ assigns to every object $A \in \mathrm{Ob}(\mathbf{C})$ an object $F(A) \in \mathrm{Ob}(\mathbf{D})$ and to every morphism $f : A \to B$ in $\mathbf{C}$ a morphism $F(f) : F(A) \to F(B)$ in $\mathbf{D}$. It must preserve composition

$$F(g \circ f) = F(g) \circ F(f)$$

and identities $F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$.

be written as $v = \sum_i \alpha_i v_i$ and the function is defined as

$$f : V \to W, \sum_i \alpha_i v_i \mapsto \sum_i \alpha_i f(v_i). \quad (5.6)$$

For example, the raising operator of a harmonic oscillator can be specified as the linear extension of

$$|n\rangle \mapsto \sqrt{n+1}\,|n+1\rangle\,.$$

A *contravariant functor* $\tilde{F}$ is similar, but reverses arrow directions, meaning

$$\tilde{F}(f) : \tilde{F}(B) \to \tilde{F}(A)$$

has the opposite direction and the order of composition is reversed in

$$\tilde{F}(g \circ f) = \tilde{F}(f) \circ \tilde{F}(g),$$

but is otherwise analogous. If unspecified, we assume by default that functors are covariant.

Given functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{C} \to \mathbf{D}$, a *natural transformation*

$$\zeta : F \Longrightarrow G$$

between them assigns to every object $A \in \mathrm{Ob}(\mathbf{C})$ a morphism $\zeta_A : F(A) \to G(A)$ in $\mathbf{D}$, such that the following diagram commutes for all $f : A \to B$ in $\mathbf{C}$;

$$\begin{tikzcd} F(A) \arrow[r, "\zeta_A"] \arrow[d, "F(f)"] & G(A) \arrow[d, "G(f)"] \\ F(B) \arrow[r, "\zeta_B"] & G(B) \end{tikzcd} \quad (5.9)$$

We can also think of "naturality" as a property of a family ζ_A of morphisms.

A *natural isomorphism* is a natural transformation ζ, for which every component ζ_A is an isomorphism.

For categories $\mathbf{C}$ and $\mathbf{D}$, the *product category* $\mathbf{C} \times \mathbf{D}$ has tuples $(A, B) \in \mathrm{Ob}(\mathbf{C}) \times \mathrm{Ob}(\mathbf{D})$ as objects and tuples $(f, g) : (A, B) \to (C, D) \in \mathbf{C}(A, C) \times \mathbf{D}(B, D)$ as morphisms, such that composition is elementwise and identities are tuples of identities.

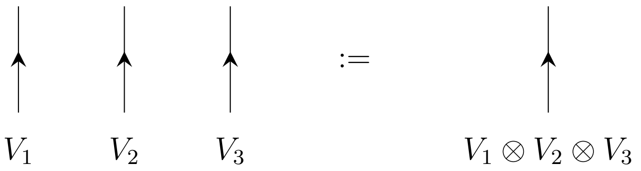
5.1.2 Tensor Product

The tensor product arises naturally in quantum mechanics. If we have two subsystems, each described by a space, the whole system is described by their tensor product. It is also the structure required to build the multi-leg tensors in tensor networks.

Given two symmetry spaces V and V', the tensor product $V \otimes V'$ is also a symmetry space. Given two symmetric maps $f : V \to W$ and $g : V' \to W'$, their tensor product is a symmetric map $f \otimes g : (V \otimes V') \to (W \otimes W')$. The tensor product is associative up to an isomorphism $\alpha_{V,W,U} : (V \otimes W) \otimes U \xrightarrow{\cong} V \otimes (W \otimes U)$. We suppress this isomorphism and will not write brackets for multiple tensor products, implying α isomorphisms as needed. There is a special symmetry space called the monoidal unit I, which can be added "for free", i.e., such that $V \otimes I \cong V \cong I \otimes V$. In tensor networks, this corresponds to adding (removing) trivial one-dimensional legs to (from) a tensor. The tensor product cooperates with map composition

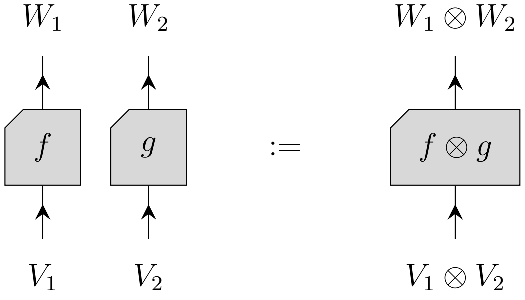
$$(f_2 \otimes g_2) \circ (f_1 \otimes g_1) = (f_2 \circ f_1) \otimes (g_2 \otimes g_1). \tag{5.10}$$

In the graphical calculus, the tensor product of spaces is represented by drawing the spaces next to each other horizontally. We do not need brackets; α isomorphisms between equivalent but unequal bracketings are implied.



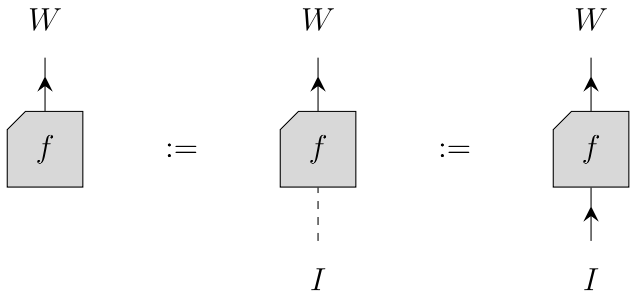
(5.11)

The tensor product of maps is drawn by drawing them next to each other.



(5.12)

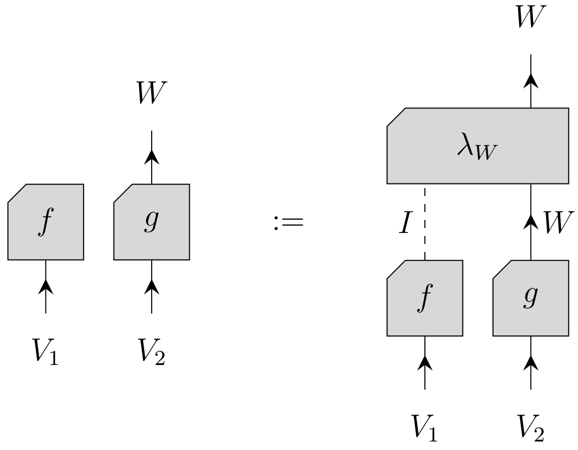
Because of (5.10), there is no ambiguity in diagrams involving both composition and tensor products of maps. The monoidal unit I is drawn as a dashed line if needed for emphasis or typically omitted altogether.



(5.13)

Omitting I from the drawing may require implicit isomorphisms, such as for example

$\lambda_W : I \otimes W \xrightarrow{\cong} W$ to be inserted.



(5.14)

There might be several ways to do this in larger diagrams. They are all equivalent, meaning the resulting maps are equal, by *coherence* theorems, see e.g. [171] or [172, chpt. 7], of the graphical calculus:

Two symmetric maps are equal if and only if their diagrams are equivalent up to planar isotopy, that is up to moving the morphisms around in the plane, or deforming the wires, up to hard-core constraints, i.e. such that morphisms or wires never touch.

We understand tensors as symmetric maps between (tensor products of) symmetric spaces, e.g.

$$T : W_1 \otimes W_2 \to V_1 \otimes V_2 \otimes V_3 \tag{5.15}$$

is a tensor with five legs. Its five legs are partitioned into the *domain* $W_1 \otimes W_2$ and *codomain* $V_1 \otimes V_2 \otimes V_3$. In the case of group symmetries, a common alternative notion is to understand the tensors as elements of tensor product space. This is fully contained in the above picture, since e.g. $t \in V_1 \otimes V_2$ is equivalent to the map $T : \mathbb{C} \to V_1 \otimes V_2, \alpha \mapsto \alpha t$, since we can recover $t = T(1)$.

Concrete case: Group Representation

The tensor product of symmetry spaces is the tensor product of vector spaces, together with an inner product defined as the bilinear extension of

$$\langle v_1 \otimes w_1 | v_2 \otimes w_2 \rangle := \langle v_1 | v_2 \rangle \langle w_1 | w_2 \rangle$$

and with the group representation

$$U_{V \otimes W} : g \mapsto U_V(g) \otimes U_W(g).$$

The tensor product of symmetric maps is the tensor product of linear maps. The resulting linear maps are indeed equivariant, i.e. qualify as symmetric maps, by con-

General case: Tensor Category

For a category $\mathbf{C}$, a monoidal structure is given by a functor $\otimes : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$, which provides the tensor product $A \otimes B$ of objects and $f \otimes g$ of morphisms. It requires the following data

- For objects A, B, C, the associator $\alpha_{ABC} : (A \otimes B) \otimes C \xrightarrow{\cong} A \otimes (B \otimes C)$.
- The monoidal unit $I \in \mathrm{Ob}(\mathbf{C})$.
- For each object A, the left unitor $\lambda_A : I \otimes A \xrightarrow{\cong} A$.

struction of the group representation on the product space.

Given orthonormal bases $\{|v_n\rangle\}_n$ as well as $\{|w_m\rangle\}_m$ of symmetry spaces V and W, an orthonormal basis for $V \otimes W$ is given by $\{|v_n\rangle \otimes |w_m\rangle\}_{n,m}$. Thus, to define a symmetric map on a tensor product by linear extension, it is enough to define the image of factorized elements of the form $|v\rangle \otimes |w\rangle$. This generalizes to nested tensor products.

The monoidal unit I is with the trivial representation $U_I : g \mapsto \mathbb{1}$ on the one-dimensional space $\mathbb{C}$. The defining isomorphisms $I \otimes A \cong A$ and $A \otimes I \cong A$ are the linear extensions of $(1 \otimes a) \mapsto a$ and $(a \otimes 1) \mapsto a$, respectively.

The tensor product is indeed associative up to an isomorphism given by linear extension of $(a \otimes b) \otimes c \mapsto a \otimes (b \otimes c)$, which is typically suppressed by omitting the brackets.

- For each object A, the right unitor $\rho_A : A \otimes I \xrightarrow{\cong} A$.

such that α, λ and ρ are natural isomorphisms and such that the triangle equation (5.16) and the pentagon equation (5.17) commute. The left unitor λ is a natural isomorphism between the functor $(I \otimes -)$ that maps A to $I \otimes A$ and f to $\mathrm{id}_I \otimes f$ and the identity functor. Similarly, the right unitor is a natural isomorphism between the functor $(- \otimes I)$ and the identity. Their existence characterizes I as the monoidal unit. The associator is a natural isomorphism between the two inequivalent double tensor product functors $((- \otimes -) \otimes -)$ and $(- \otimes (- \otimes -))$, both $\mathbf{C} \times \mathbf{C} \times \mathbf{C} \to \mathbf{C}$.

The consistency conditions for the monoidal structure are that the triangle equation

$$\begin{tikzcd}
(A \otimes I) \otimes B \arrow[rr, "\alpha_{AIB}"] \arrow[dr, "\rho_A \otimes \mathrm{id}_B"] & & A \otimes (I \otimes B) \arrow[dl, "\mathrm{id}_A \otimes \lambda_B"] \\
& A \otimes B &
\end{tikzcd} \tag{5.16}$$

and the pentagon equation

$$\begin{tikzcd}
(A \otimes (B \otimes C)) \otimes D \arrow[rr, "\alpha_{A,B\otimes C,D}"] & & A \otimes ((B \otimes C) \otimes D) \arrow[d, "\mathrm{id}_A \otimes \alpha_{BCD}"] \\
((A \otimes B) \otimes C) \otimes D \arrow[u, "\alpha_{ABC} \otimes \mathrm{id}_D"] \arrow[dr, "\alpha_{A\otimes B,C,D}"] & & A \otimes (B \otimes (C \otimes D)) \\
& (A \otimes B) \otimes (C \otimes D) \arrow[ur, "\alpha_{A,B,C\otimes D}"] &
\end{tikzcd} \tag{5.17}$$

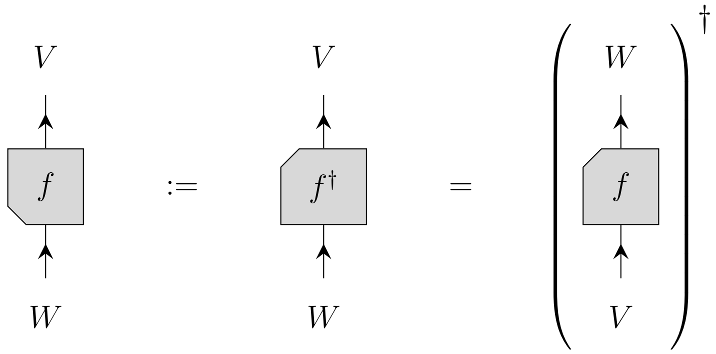
both commute.

5.1.3 Dagger

For a symmetric map $f : V \to W$, the dagger (or adjoint) is a symmetric map $f^\dagger : W \to V$ with opposite direction. Note that the dagger of a map is always composable with the original map in either order.

The dagger is the operation that allows us to form inner products on a tensor network level. Expectation values $\langle\phi|O|\phi\rangle =: \langle\phi|\tilde{\phi}\rangle$ are given in terms of the inner product on the many-body Hilbert space. If $|\phi\rangle$ is a TNS with tensors $\{B_i\}$, the inner product $\langle\phi|\tilde{\phi}\rangle$ is given as the contraction of a tensor network, where the "half" representing $\langle\phi|$ consists of the adjoint (daggered) tensors $\{B_i^\dagger\}$.

In the graphical calculus, the dagger of a map is drawn by vertically mirroring the box.



(5.18)

Note that it has the *same* label f as the original map and is only identified as its dagger because of the mirrored box. For readability, we do not mirror the label itself but rather choose a chamfered box that makes the mirroring apparent. Since the dagger reverses the order of composition but preserves the order of the tensor product, the dagger of a composite diagram is obtained graphically by first mirroring along a horizontal axis, then flipping back all arrows to their original direction.

The dagger preserves the order of tensor products $(f\otimes g)^\dagger = f^\dagger\otimes g^\dagger$, but reverses the order of composition $(f\circ g)^\dagger = g^\dagger\circ f^\dagger$. This motivates the graphical representation as a vertical mirroring and implies that a composite diagram is the dagger of another diagram if and only if they are each others vertical mirror images, up to planar isotopy.

A symmetric map is *unitary*, if its adjoint is its inverse, that is if $f^\dagger\circ f = \mathrm{id}_V$ and $f\circ f^\dagger = \mathrm{id}_W$.

Concrete case: Group Representation

The dagger is defined as usual, where the dagger of a map $f : A \to B$ is the unique map $f^\dagger : B \to A$ that fulfills

$$\langle w|f(v)\rangle = \left\langle f^\dagger(w)\middle|v\right\rangle$$

for all $v \in A$ and $w \in B$, or in braket operator notation $\langle w|f|v\rangle = \overline{\langle v|f^\dagger|w\rangle}$. We can therefore read off that its matrix representation $(f^\dagger)_{mn} = \langle m|f^\dagger|n\rangle = \overline{f_{nm}}$ is the hermitian conjugate matrix. It remains to check that the linear map $f^\dagger$ is indeed a symmetric map. Let $g \in G$, then we have $U_B(g^{-1})\circ f = f\circ U_A(g^{-1})$ since f is sym-

General case: Tensor Category

For a category $\mathbf{C}$, a dagger structure is given by a contravariant functor $\dagger : \mathbf{C} \to \mathbf{C}$ that acts as the identity on objects and fulfills

- For $f : A \to B$, the dagger $f^\dagger : B \to A$ has reversed direction (contravariant)
- For morphisms f, g, we have $(f\circ g)^\dagger = g^\dagger\circ f^\dagger$ (contravariant)
- For morphisms f, we have $(f^\dagger)^\dagger = f$
- The identities $\mathrm{id}_A^\dagger = \mathrm{id}_A$ are invariant

metric. Taking the dagger and using that the representations $U_{A/B}$ are unitary, we find that $f^\dagger$ is symmetric as well.

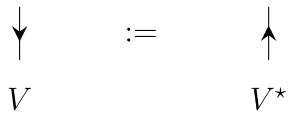
The dagger distributes over the tensor product $(f \otimes g)^\dagger = f^\dagger \otimes g^\dagger$, by definition of the scalar product on the product space. The dagger reverses the order of map composition $(f \circ g)^\dagger = g^\dagger \circ f^\dagger$, which we obtain by applying the defining property twice; $\langle w|f(g(v))\rangle = \langle g^\dagger(f^\dagger(w))|v\rangle$.

The dagger structure is compatible with a tensor product $\otimes$ if

- The action on morphisms cooperates; that is for morphisms f, g, we have $(f \otimes g)^\dagger = f^\dagger \otimes g^\dagger$
- The isomorphisms of the monoidal structure are unitary; $\alpha^\dagger_{ABC} = \alpha^{-1}_{ABC}$ and $\lambda^\dagger_A = \lambda^{-1}_A$ and $\rho^\dagger_A = \rho^{-1}_A$

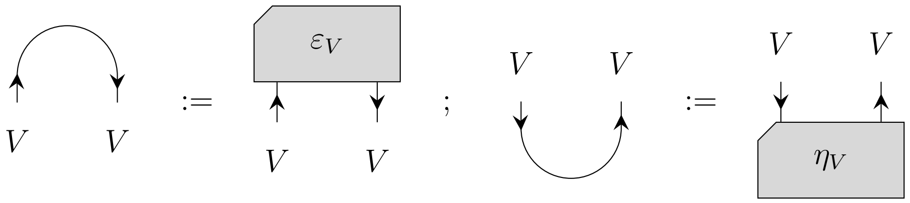
5.1.4 Duality

A duality structure is very familiar to anyone working with quantum physics; A bra vector is the dual of a ket vector. Duality introduces the concepts of dual spaces, the transpose of a map, the trace, and the quantum dimension. Graphically, duality is visualized as 180° in-plane rotation. A wire with a downward arrow represents the dual space. Note that it is labeled by the original space that it is the dual of.



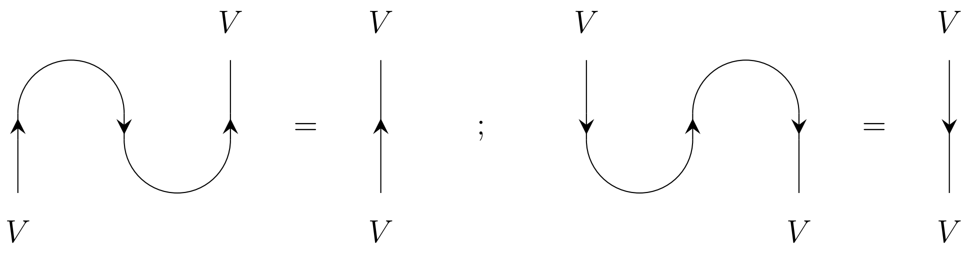
(5.19)

There is a pair of special symmetric maps related to the duality of V and $V^\star$, the *cap* $\varepsilon_V : V \otimes V^\star \to I$ and *cup* $\eta_V : I \to V^\star \otimes V$. Note the different orders of the tensor product. They are drawn as bent lines.



(5.20)

Note that the wire has a consistent arrow direction through the bend, which is why the cup and cap are precisely what characterizes $A^\star$ as the dual of A. They fulfill the snake equations

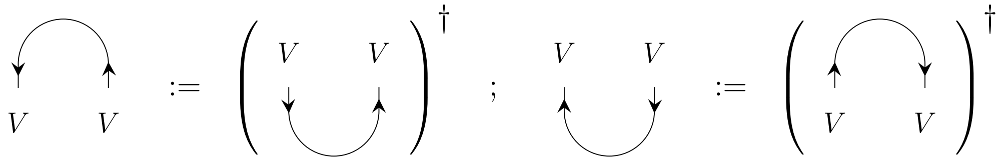


. (5.21)

As an instructive example for reading these graphical representations, let us explictly write out the first equality. Including all implied isomorphisms, it reads

$$\lambda_V \circ (\varepsilon_V \otimes \mathrm{id}_V) \circ \alpha_{VV^\star V} \circ (\mathrm{id}_V \otimes \eta_V) \circ \rho_V^\dagger = \mathrm{id}_V \ . \tag{5.22}$$

The dagger gives us the "opposite cap" $\eta_V^\dagger : V^\star \otimes V \to I$ and "opposite cup" $\varepsilon_V^\dagger : I \to V \otimes V^\star$.



(5.23)

Note the different arrow directions compared to the regular cup and cap.

Taking the dual of a tensor product results in the tensor product of individual duals in reverse order, up to a unitary isomorphism $\zeta_{V,W}^{-1} : (V \otimes W)^\star \xrightarrow{\cong} W^\star \otimes V^\star$, given explicitly in (5.29). This isomorphism is implied in the graphical language; if we rotate a tensor product $V \otimes W$, we graphically obtain $W^\star \otimes V^\star$. Thus, in graphical equations, we may simply use $W^\star \otimes V^\star$ as a dual of $V \otimes W$.

Taking the dual twice gives the same space back; $V^{\star\star} \cong V$, again up to an isomorphism $\pi_V : V \to V^{\star\star}$, given explicitly in (5.28). This isomorphism is also implied by the graphical language, as rotating twice gives us back the same diagram, not a double dual. Thus, we never need to use double duals in the graphical notation and can always use V as a dual of $V^\star$.

The monoidal unit $I = I^\star$ is self-dual where the cups are given by the unitors $\varepsilon_I = \eta_I^\dagger = \rho_I = \lambda_I$ and the caps by their daggers. This allows us to omit the arrows on the (dashed) wires for the monoidal unit.

Concrete case: Group Representation

For a symmetry space V with a group representation U_V, the dual space is the dual vector space $V^\star = \{\langle\phi| : V \to \mathbb{C} | \langle\phi| \text{ linear}\}$ with the contragradient representation $U_{V^\star}$, that is

$$U_{V^\star}(g) : \langle\phi| \mapsto \langle\phi|\, U_V^\dagger(g). \tag{5.24}$$

Where the notation above means that the image of the dual vector $|\psi\rangle \mapsto \langle\phi|\psi\rangle$ under the representation is the dual vector $|\psi\rangle \mapsto \langle\phi|U_V^\dagger(g)|\psi\rangle$.

The cup is given by

$$\eta_V : \mathbb{C} \to V^\star \otimes V, \alpha \mapsto \alpha \sum_n \langle\varphi_n| \otimes |\varphi_n\rangle \ , \tag{5.25}$$

General case: Tensor Category

In a category $\mathbf{C}$ with tensor product $\otimes$, which has a dagger, we define a duality structure[1] as the following data;

- For every object $A \in \mathrm{Ob}(\mathbf{C})$, a *dual* object $A^\star$
- For every object A, two morphisms: the cup $\eta_A : I \to A^\star \otimes A$ and the cap $\varepsilon_A : A \otimes A^\star \to I$

[1] We have taken quite the shortcut here, compared to common literature, e.g. compared to [166, chpt 3], by implicitly defining a pivotal structure given by π such that the duals are dagger duals.

where $\{|\varphi_n\rangle\}$ is an orthonormal basis of V and $\{\langle\varphi_n|\}$ the associated orthonormal dual basis of $V^\star$. The cap is given by

$$\varepsilon_V : V \otimes V^\star \to \mathbb{C}, |\psi\rangle \otimes \langle\phi| \mapsto \langle\phi|\psi\rangle \quad (5.26)$$

The isomorphism between a symmetry space V and its double-dual $V^{\star\star}$ is given by

$$\pi_V : V \to V^{\star\star}, |\psi\rangle \mapsto (\langle\phi| \mapsto \langle\phi|\psi\rangle)\,, \quad (5.27)$$

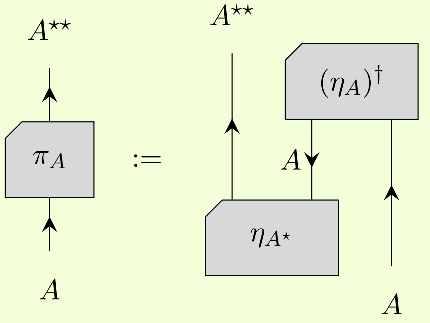
where the expression in round brackets is a double-dual vector, as it linearly maps a dual (bra) vector to a complex number.

Given dual vectors $\langle v| \in V^\star$ and $\langle w| \in W^\star$, their tensor product $\langle v| \otimes \langle w| \in V^\star \otimes W^\star$ is isomorphic to a dual vector in $(W \otimes V)^\star$ that takes the form $|\psi\rangle \mapsto \langle w \otimes v|\psi\rangle$. The reversed order in $(W \otimes V)^\star \cong V^\star \otimes W^\star$ is an arbitrary choice here, but is natural in the graphical language, as bending lines like in equation (5.29) reverses the order.

such that the snake equations (5.21) are fulfilled.

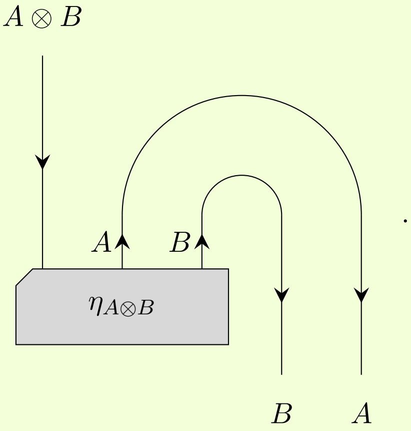
Taking the dagger yields the opposite cup $\varepsilon_A^\dagger$ and opposite cap $\eta_A^\dagger$, which witness $A^\star$ as a left dual of A and inherit analogous snake equations, given by the dagger of (5.21). Since $A^\star$ is both a left and a right dual of A and is the canonical choice among possibly multiple duals of A, it is simply referred to as *the* dual of A.

It also makes the notion of duality reflexive, that is, A is a dual object of $A^\star$. It may, however, not agree with the choice $A^{\star\star}$ for *the* dual of $A^\star$. They are isomorphic by an isomorphism $\pi_A : A \to A^{\star\star}$, defined as the following composite.



(5.28)

The dual $(A \otimes B)^\star$ of a tensor product is isomorphic to $B^\star \otimes A^\star$ by a unitary isomorphism $\zeta_{A,B} : B^\star \otimes A^\star \xrightarrow{\cong} (A \otimes B)^\star$, defined as the following composite

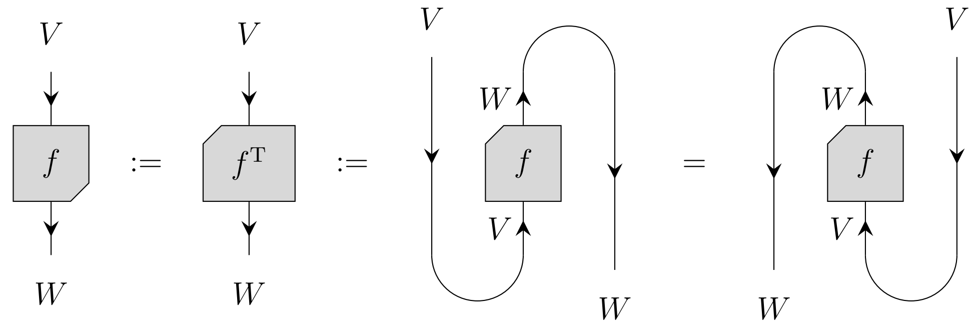


(5.29)

Note the similarity to the construction of the π isomorphism. This generalizes; different choices for the dual of a given object are isomorphic, where the isomorphism can be constructed as the “mixed snake” between a cup from one duality and a cap from the other.

It turns out that taking duals is functorial if we understand the duals functor to act as transposition (5.30) on morphisms.

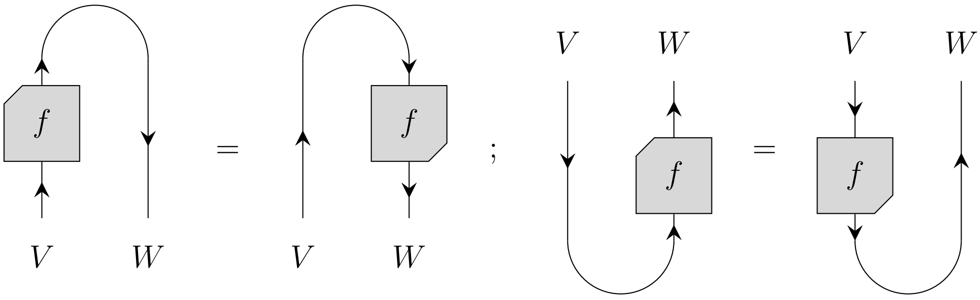
Given a symmetric map $f : V \to W$, its *transpose* $f^{\mathrm{T}} : W^\star \to V^\star$ is another symmetric map. It is defined as the following composite of a cup, f itself, and a cap.



(5.30)

The alternative definition with opposite cap and cup is equal. Taking the transpose inverts the order of composition $(f \circ g)^{\mathrm{T}} = g^{\mathrm{T}} \circ f^{\mathrm{T}}$, which follows from the snake equation (5.21) and is visually intuitive in the graphical calculus, where the rotation inverts the vertical order. Another consequence of the snake equation is that $(f^{\mathrm{T}})^{\mathrm{T}} = f$, as well as $\mathrm{id}_V^{\mathrm{T}} = \mathrm{id}_{V^\star}$.

The sliding properties



(5.31)

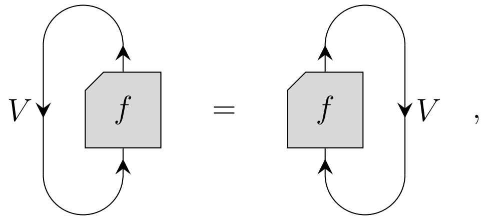
for all $f : V \to W$ also follow from the snake rules and can be visualized as sliding the box along the wire. The rotation induced by the bend means that the transpose appears on the other side. Analogous properties for sliding along the opposite cup and cap, not explicitly shown here, also hold.

The graphical calculus, including duality, also fulfills coherence theorems. Two symmetric maps formed from the building blocks introduced so far are equal if and only if their diagrams are equivalent up to oriented planar isotopy. Note the new qualifier “oriented”, which means that arrow directions must remain consistent if the lines are bent, as e.g. in the snake rules (5.21). Additionally, two morphisms given as composite diagrams are each others transpose if and only if one diagram is the rotated version of the other, up to oriented planar isotopy.

As the notation suggests, the opposite cup (cap) is also equal to the transpose of the regular cap (cup), up to suppressed isomorphisms, e.g. $(\varepsilon_V)^{\mathrm{T}} : I^\star \to (V^\star \otimes V)^\star$ is equal to the following composite

$$I^\star = I \xrightarrow{\varepsilon_V^\dagger} V \otimes V^\star \xrightarrow{\pi_V \otimes \mathrm{id}_{V^\star}} V^{\star\star} \otimes V^\star \xrightarrow{(\zeta_{V^\star,V})^{-1}} (V^\star \otimes V)^\star.$$

Another relevant composite object is the *trace* $\mathrm{Tr}\,(f)$ of a symmetric map $f : V \to V$, defined as

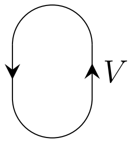
$\mathrm{Tr}\,(f) \quad :=$  $\quad , \qquad (5.32)$

where the alternative definition, which closes the loop to the right-hand side, is equal. The trace is invariant under transposition $\mathrm{Tr}\left(f^{\mathrm{T}}\right) = \mathrm{Tr}\,(f)$ by the sliding property (5.31), and is cyclic

$$\mathrm{Tr}\,(f \circ g) = \mathrm{Tr}\,(g \circ f)\,. \tag{5.33}$$

Note that the trace is a symmetric map $I \to I$. We identify a one-to-one correspondence of such maps with complex numbers in subsection 5.1.8, which reconciles this definition with the usual trace of linear maps, which is a number. Let us briefly already assume that we can treat it as a number.

The *quantum dimension* of a space is defined as

$\dim V \quad := \quad \mathrm{Tr}\,(\mathrm{id}_V) \quad =$  $\qquad (5.34)$

and we find that $\dim V^\star = \dim V$, i.e. a loop with opposite arrow direction agrees with the loop above. For a group symmetry, the quantum dimension coincides with the vector space dimension of the symmetry space and is, in particular, always an integer. In general, we can observe that it is invariant under the dagger and thus real. We find it is non-negative for all symmetries we use in practice.

Finally, the trace and dagger induce the Frobenius *inner product* $\langle f|g\rangle_{\mathrm{F}} = \mathrm{Tr}\left(f^\dagger \circ g\right)$ and Frobenius *norm* $\|f\|_{\mathrm{F}} = \sqrt{\langle f|f\rangle_{\mathrm{F}}}$ of symmetric maps.

5.1.5 Braids

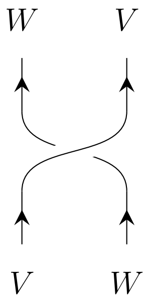
We have already seen that the tensor product is associative, at least up to isomorphism. The next natural question to ask is if it is also commutative. This is related to the question if (and how) legs on a tensor can be swapped/permuted.

We assume that an isomorphism

$$\tau_{V,W} : V \otimes W \overset{\cong}{\to} W \otimes V \tag{5.35}$$

called the *braid*, exists for every pair V, W of symmetry spaces, such that the tensor product is indeed commutative up to isomorphism. While that isomorphism is straightforward in the case of group symmetries, it needs to be carefully kept track of in the case of fermionic or anyonic grading, where it captures the exchange statistics. Graphically, we draw the standard over-braid as the following crossing

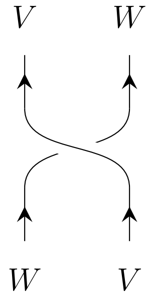
$\tau_{V,W} \quad =:$



(5.36)

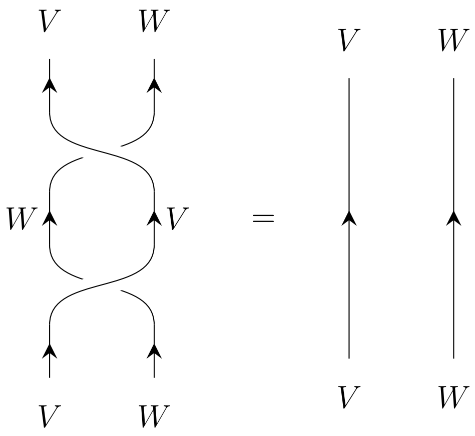
of wires. Viewed from a point of view from the bottom of the diagram, looking upward, it is a clockwise rotation of the wires. The inverse braid, or "under-braid", has opposite chirality and is drawn as such.

$(\tau_{V,W})^{-1} \quad =:$



(5.37)

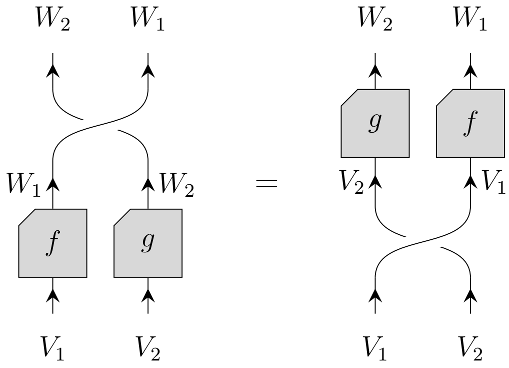
As the graphical notation suggests, the braid is unitary and $(\tau_{V,W})^{-1} = (\tau_{V,W})^{\dagger}$. It is visually intuitive that an under-braid undoes an over-braid.



(5.38)

Note that the two ways of braiding $V \otimes W \to W \otimes V$ are $\tau_{V,W}$ and $(\tau_{W,V})^{\dagger}$, with opposite subscripts. They are different in general, but if they are equal, we call the braid *symmetric*.

The braid fulfills the following *sliding property*.



(5.39)

Concrete case: Group Representation

For group symmetries, the braid $\tau_{V,W}$ is the linear extension of

$$|v\rangle \otimes |w\rangle \mapsto |w\rangle \otimes |v\rangle \,.$$

It is, in particular, always symmetric. The properties stated above are straight-forward to verify.

General case: Tensor Category

For a category $\mathbf{C}$ with tensor product $\otimes$, a braiding structure is given by a unitary natural isomorphism $\tau_{A,B} : A\otimes B \to B\otimes A$, the *braid*, which fulfills the hexagon equations (5.40) and (5.41). It is a natural isomorphism between the tensor product functor $\otimes$ and a "reverse tensor product" $\mathbf{C} \times \mathbf{C} \to \mathbf{C}$ that assigns to (A, B) the object $B\otimes A$ and to (f, g) the morphism $g\otimes f$. Thus, naturality is equivalent to the sliding property (5.39).

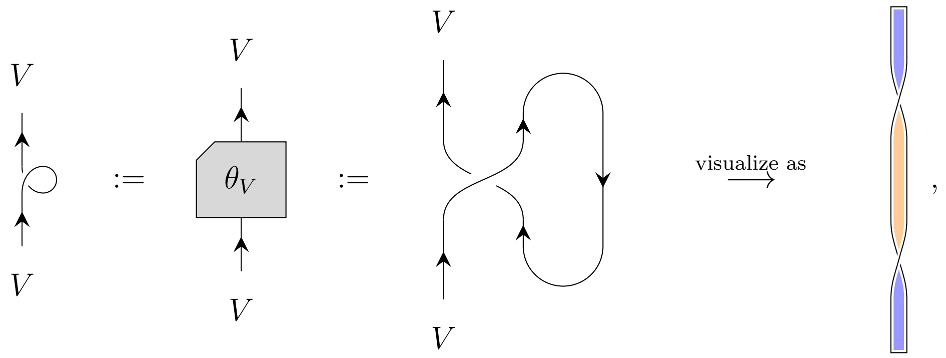
The braid fulfills the following consistency conditions, namely that the so called *hexagon equations* commute;

$$\begin{tikzcd}
& A \otimes (B \otimes C) \arrow[r, "\tau_{A,B\otimes C}"] \arrow[dl, "\alpha^{-1}_{A,B,C}"] & (B \otimes C) \otimes A & \\
(A \otimes B) \otimes C \arrow[dr, "\tau_{A,B}\otimes \mathrm{id}_C"] & & & B \otimes (C \otimes A) \arrow[ul, "\alpha^{-1}_{B,C,A}"'] \\
& (B \otimes A) \otimes C \arrow[r, "\alpha_{B,A,C}"] & B \otimes (A \otimes C) \arrow[ur, "\mathrm{id}_B\otimes\tau_{A,C}"] &
\end{tikzcd} \tag{5.40}$$

$$\begin{tikzcd}
& (A \otimes B) \otimes C \arrow[r, "\tau_{A\otimes B,C}"] \arrow[dl, "\alpha_{A,B,C}"] & C \otimes (A \otimes B) & \\
A \otimes (B \otimes C) \arrow[dr, "\mathrm{id}_A\otimes\tau_{B,C}"] & & & (C \otimes A) \otimes B \arrow[ul, "\alpha_{C,A,B}"'] \\
& A \otimes (B \otimes C) \arrow[r, "\alpha^{-1}_{A,C,B}"] & (A \otimes C) \otimes B \arrow[ur, "\tau_{A,C}\otimes \mathrm{id}_B"] &
\end{tikzcd} \tag{5.41}$$

The braids introduce a third dimension to the graphical notation. With braids, the coherence theorem is that two symmetric maps are equal if and only if their diagrams are equivalent up to the spatial isotopy of ribbons. That means diagrams may be deformed in a three-dimensional ambient space as long as the order of the endpoints of the open wires remains fixed and wires and morphisms do not touch each other. Additionally, we need to think of the wires as having some finite width, i.e., as ribbons, where the endpoints are not only fixed in space but also need to have fixed rotation around the wire axis, such that a twist in the ribbon can not be resolved as a part of the isotopy.

Such a twist is captured by a map that is defined as the following composite



(5.42)

where we can understand its graphical representation as a miniature of the definition.

It fulfills the following defining property

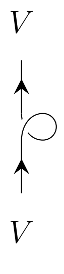
$$\theta_{V\otimes W} = (\theta_V \otimes \theta_W) \circ \tau_{W,V} \circ \tau_{V,W} \ . \tag{5.43}$$

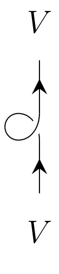
In the three-dimensional isotopy, we may think of the twist as literally a twist in a ribbon, as depicted on the very right of equation (5.42). We found it instructive to confirm this relation with a physical ribbon, e.g. a thin stripe of paper, by forming the configuration on the LHS, then pulling the ends tight.

We can get three more twist-like maps $V \to V$ by either taking the dagger or by taking the transpose and substituting $V^\star$ for V, or both. They all appear in equation (5.45). We assume

$$(\theta_A)^{\mathrm{T}} = \theta_{A^\star} \ , \tag{5.44}$$

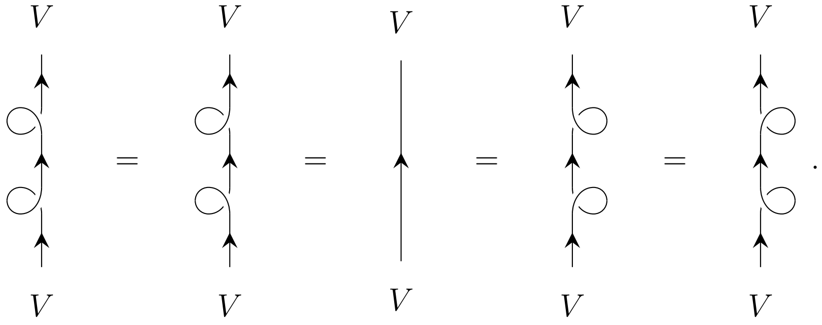
which implies that they are related as follows

θ_V = 

= 

= $(\theta_{V^\star})^{\mathrm{T}}$, (5.45) which we can visualize as a 3D rotation around a vertical axis or as “folding over”.

The twist is unitary, meaning it is undone by its dagger, which gives the following relations



(5.46)

Again, we find it instructive to confirm these relations, as well as (5.45) with physical ribbons.

5.1.6 Linear structure

The symmetric maps have a linear structure, meaning the space $\mathrm{Hom}\,(V,\,W)$ of symmetric maps between symmetry spaces V, W is a complex vector space, such that we can form linear combinations of symmetric maps. The following conditions on the linear structure are natural if we think of linear maps between vector spaces as the prototype for symmetric maps. The linear structure cooperates with map composition, such that $(f, g) \mapsto f \circ g$ is bilinear, meaning

$$\begin{aligned} f \circ (ag + bg') &= a(f \circ g) + b(f \circ g') \\ (af + bf') \circ g &= a(f \circ g) + b(f' \circ g) \end{aligned} \tag{5.47}$$

for maps $f, f' : V' \to W$, $g, g' : V \to V'$ and scalars $a, b \in \mathbb{C}$.

The linear structure similarly cooperates with the tensor product, such that $(f, g) \mapsto f \otimes g$ is bilinear,

$$\begin{aligned} f \otimes (ag + bg') &= a(f \otimes g) + b(f \otimes g') \\ (af + bf') \otimes g &= a(f \otimes g) + b(f' \otimes g) \end{aligned}. \tag{5.48}$$

The linear structure cooperates with the dagger if the dagger is antilinear

$$(af + bf')^\dagger = \overline{a} f^\dagger + \overline{b}(f')^\dagger, \tag{5.49}$$

where $\overline{a}$ denotes the complex conjugate of a.

Concrete case: Group Representation

For linear maps, the vector space structure is straight-forward, where linear combina-

General case: Tensor Category

In category theory, the most straightforward way to define a linear structure is via tions of maps $f, g : V \to W$ with coefficients $a, b \in \mathbb{C}$ are given by

$$(af + bg) : V \to W, x \mapsto af(x) + bg(x).$$

It remains to check that the linear combination is a symmetric map, which follows directly, as the group representation is also linear. As a conclusion, the set $\mathrm{Hom}\,(V,\ W)$ of symmetric maps between symmetry spaces V, W is indeed a complex vector space.

the notion of a linear category. In particular, a $\mathbb{C}$-linear category is a category $\mathbf{C}$ where all collections of morphisms $\mathbf{C}(A, B)$ are vector spaces over $\mathbb{C}$, such that map composition is bilinear (5.47). In particular, this means that there is a zero morphism $0_{A,B} : A \to B$ for every pair A, B of objects, the zero vector of the vector space. Since composition is bilinear, we have $0_{B,C} \circ f = 0_{A,C} = g \circ 0_{A,B}$ for all $f : A \to B$ and $g : B \to C$. The linear structure cooperates with the monoidal or dagger structures, respectively, if the compatibility axioms (5.48) or (5.49) are fulfilled.

5.1.7 Direct sums

The next structure we want to introduce formalizes the idea that a symmetry partitions Hilbert spaces into sectors according to quantum numbers. In this section, we define the direct sum, which allows us to understand how to build up larger spaces from smaller building blocks. We go in the opposite direction and ask if a given space can be deconstructed and what the elementary building blocks are in the next section.

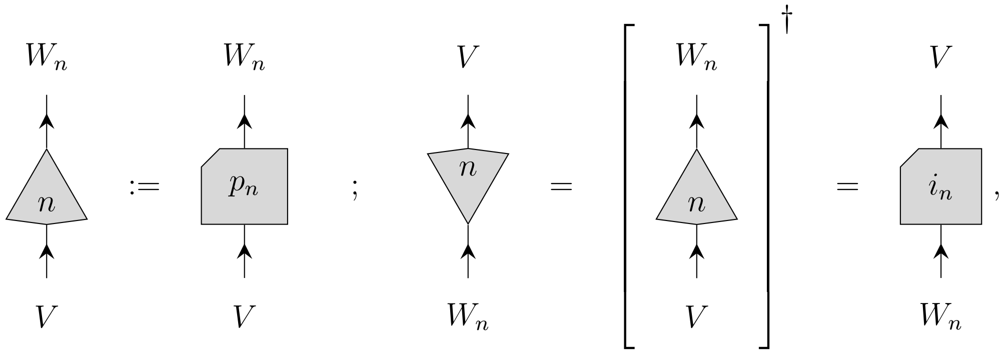
We define the *direct sum* $V = W_1 \oplus W_2 \oplus \cdots \oplus W_N = \bigoplus_{n=1}^{N} W_n$ of symmetry spaces $W_1, W_2, \ldots, W_N$ in the columns below. If we think of elements of the spaces as column vectors, the elements of the direct sum are vertically stacked column vectors. It is characterized by symmetric *projection* maps $p_n : V \to W_n$ that tell us how to pick the components that belong to W_n from the vertical stack. The related symmetric *injection* maps $i_n : W_n \to V$ tell us how to embed components from a vector in W_n into the larger space and are related to the projections via the dagger $i_n = p_n^\dagger$. They are orthonormal $p_n \circ i_m = \delta_{m,n}\mathrm{id}_{W_n}$ and complete $\sum_n i_n \circ p_n = \mathrm{id}_V$.

Here, we employ a slight abuse of notation regarding the Kronecker delta since for $m \neq n$, the orthonormality equation is (in general) ill-typed and has type $W_m \to W_n$ on the LHS but $W_n \to W_n$ on the RHS. We understand the Kronecker delta to fulfill laxly the following role for any map f

$$\delta_{m,n} f := \begin{cases} f & m = n \\ 0 \quad \text{(zero map of correct type in the given context)} & m \neq n \end{cases}, \tag{5.50}$$

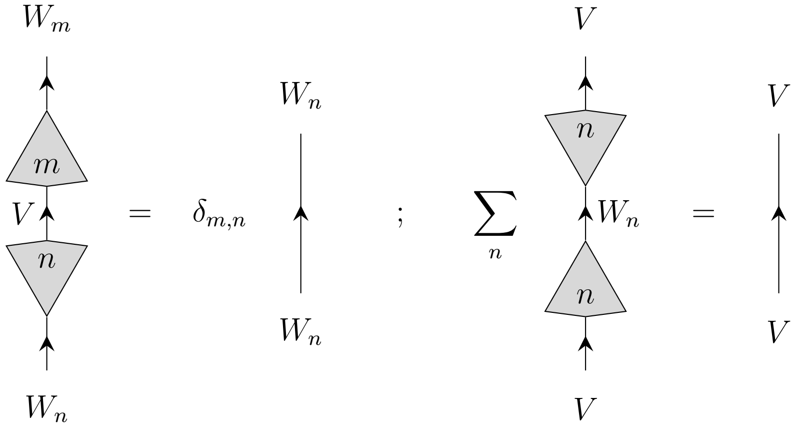
where the “correct type in the given context” may not be the type of f. Note that we even use this notation if the expression for f is ill-defined in the $m \neq n$ case.

We denote the injection and projection maps in the graphical calculus as kites



(5.51)

where the kite “points” from the larger space V to the smaller space $W_{n/m}$. Orthonormality and completeness take the following form.



(5.52)

If we find some other space $\tilde{V}$ with projections $\tilde{p}_n$ and inclusions $\tilde{i}_n$ that also fulfill (5.52), we can conclude that $\tilde{V} \cong \bigoplus_n W_n$ is isomorphic to the direct sum, by a unitary isomorphism $\sum_n i_n \circ \tilde{p}_n$.

Note that the graphical notation for inclusions and projections is horizontally symmetric, such that we can not distinguish dagger and transpose graphically. We resolve this by using the transposed projections $p_n^{\mathrm{T}} : W_n^\star \to V^\star$ as the inclusions of $V^\star \cong \bigoplus_n W_n^\star$ and similarly the transposed inclusions as projections.

The tensor product distributes over direct sums, up to isomorphism, that is

$$V \otimes (W \oplus W') \cong (V \otimes W) \oplus (V \otimes W') \tag{5.53}$$

$$(V \oplus V') \otimes W \cong (V \otimes W) \oplus (V' \otimes W) \ . \tag{5.54}$$

We can see this directly since $\mathrm{id}_V \otimes p_n$ is a projection and $\mathrm{id}_V \otimes i_n$ an inclusion for the first direct sum, if p_n (i_n) is a projection (inclusion) for $W \oplus W'$, and analogously for the second case.

Concrete case: Group Representation

The direct sum $V = \bigoplus_n W_n$ of symmetry spaces is defined as follows. It is the direct sum of vector spaces, which is the set

$$\bigoplus_n W_n = \{(w_1, \dots, w_N) | w_n \in W_n\}$$

together with elementwise addition and elementwise scalar multiplication. The scalar product is also defined elementwise, that is

$$\langle (w_1, \dots, w_N) | (v_1, \dots, v_N) \rangle = \prod_n \langle w_n | v_n \rangle ,$$

as is the group representation

$$U_V(g) = \bigoplus_n U_{W_n}(g).$$

Here, the direct sum of maps denotes elementwise application, e.g.

$$f \oplus g : (v, w) \mapsto (f(v), g(w))$$

for a binary direct sum with straight-forward generalization to N-ary sums.

The injection maps are given by

$$i_n : W_n \to V, v_n \mapsto (0, \dots, v_n, \dots, 0),$$

where v_n sits at the n-th position in the tuple. The projection maps are

$$p_n : V \to W_n (v_1, \dots, v_n, \dots, v_N) \mapsto v_n.$$

Orthonormality and completeness are easy to check.

General case: Tensor Category

The direct sum[2] of objects $A_1, \dots, A_N$ is an object $B := \bigoplus_n A_n$ equipped with the inclusions $i_n : A_n \to B$ and projection $p_n : B \to A_n$ morphisms for $n = 1, \dots, N$, which are orthonormal and complete (5.52).

We assume that our category $\mathbf{C}$ has a direct sum $\bigoplus_{n=1}^{N} A_n \in \mathrm{Ob}(\mathbf{C})$ for any finite set $\{A_1, \dots, A_N\}$ of objects.

The duality structure guarantees that $\otimes$ distributes over $\oplus$, in the sense of equation (5.53), see e.g. [166, Sec. 3.3] for a derivation of the isomorphisms.

The direct sums are compatible with a dagger structure if $i_n = p_n^\dagger$.

[2] This is also known as a biproduct in the literature.

5.1.8 Sectors

In this section, we introduce the elementary building blocks – the simple spaces. In the following, we characterize them as building blocks in the sense that all symmetry spaces can be written as direct sums of them, meaning any symmetry space is equal or at least isomorphic to a direct sum of simple spaces. Additionally, we characterize them as elementary in the sense that they themselves admit no further decomposition into non-trivial direct sums.

The most direct handle on whether a space V can be decomposed into a direct sum is the dimension of its endomorphism space $\mathrm{End}\,(V)$. Consider the following linear map between vector spaces

$$\phi : \mathrm{End}\left(\bigoplus_n W_n\right) \to \hat{\bigoplus_n} \mathrm{End}\,(W_n)\,, f \mapsto \hat{\bigoplus_n} p_n \circ f \circ i_n, \tag{5.55}$$

where i_n (p_n) are the injections (projections) of the direct sum $\bigoplus_n W_n$. Note that we write a little hat on top to distinguish the direct sum $\hat{\oplus}$ of vector spaces from the direct sum $\oplus$ of symmetry spaces. We can conclude that ϕ is surjective, since $\hat{\bigoplus}_n f_n \mapsto \sum_n i_n \circ f_n \circ p_n$ is a right inverse. Therefore, we get $\dim \mathrm{End}\,(\bigoplus_n W_n) \geq \sum_n \dim \mathrm{End}\,(W_n)$, i.e. on taking direct sums, the dimension of the endomorphism space increases at least additively[1].

We count symmetry spaces with zero-dimensional endomorphism spaces as trivial. Thus, we can identify a class of elementary spaces; A *simple* symmetry space V is a symmetry space with a one-dimensional endomorphism space $\mathrm{Hom}\,(V,\ V)$. It remains to argue that the simple spaces are the building blocks for symmetry spaces, i.e. that any symmetry space is (equivalent to) a direct sum of simple spaces. For a group symmetry, this can be guaranteed if the group representations are unitary, which we assume. For a general category, we include it as a requirement for being a tensor category.

Towards a classification of the simple spaces, note[2] that two isomorphic spaces $W \cong \tilde{W}$ are interchangeable in direct sums, that is, $V \oplus W \cong V \oplus \tilde{W}$, and straightforward generalizations to direct sums of many spaces. Thus, one representative per isomorphism class of simple spaces is enough to build any symmetry space via direct sum. These representatives are the *sectors*, which fulfill the following properties.

- There is a set $\mathcal{S}$ of symmetry spaces, the *sectors*, that is either finite or countably infinite.
- The monoidal unit $I \in \mathcal{S}$ is a sector. We call it the *trivial sector*.
- The space $\mathrm{End}\,(a)$ of symmetric maps from a sector $a \in \mathcal{S}$ to itself is one-dimensional.
- The space $\mathrm{Hom}\,(a,\ b)$ of symmetric maps between distinct sectors $a, b \in \mathcal{S}$ with $a \neq b$ is zero-dimensional.
- Every symmetry space V is isomorphic to a finite direct sum of sectors, meaning there is an integer $N_a^V \in \mathbb{N}_0$ for every sector a such that (5.57) holds, while $\sum_{a\in\mathcal{S}} N_a^V < \infty$ is finite.

[1] If we already use the properties of sectors derived/assumed in the following, we can conclude that the dimension is additive if the W_n do not share any sectors, meaning there is no sector a for which more than one $N_a^{W_n}$ is non-zero. If they do share sectors, it is strictly larger than additive. For example, for a sector a we have $\dim \mathrm{End}\,(a \oplus a) = 4$ and $\dim \mathrm{End}\,(a) = 1$.

[2] Let us sketch a proof. If the assumed isomorphism is $\phi : W \xrightarrow{\cong} \tilde{W}$, we find an isomorphism $i_V^{V\oplus\tilde{W}} \circ p_V^{V\oplus W} + i_{\tilde{W}}^{V\oplus\tilde{W}} \circ \phi \circ p_W^{V\oplus W}$ that establishes the claim. Its inverse is built analogously, exchanging $W \leftrightarrow \tilde{W}$ and $\phi \leftrightarrow \phi^{-1}$.

The properties imply for any pair V, W of simple spaces

$$\dim \mathrm{Hom}\,(V,\ W) = \begin{cases} 1 & V \cong W \\ 0 & \text{else} \end{cases}. \tag{5.56}$$

To see this, let $\phi_V : V \overset{\cong}{\to} a$ and $\phi_W : W \overset{\cong}{\to} b$ be the isomorphisms to sectors $a, b \in \mathcal{S}$. Then, $f \mapsto \phi_W \circ f \circ \phi_V^{-1}$ is a vector space isomorphism, which establishes $\dim \mathrm{Hom}\,(V,\ W) = \dim \mathrm{Hom}\,(a,\ b) = \delta a, b$.

Writing out the last property in the list above gives us the *sector decomposition*

$$V \cong \bigoplus_{a \in \mathcal{S}} \bigoplus_{\mu=1}^{N_a^V} a \tag{5.57}$$

for any symmetry space V.

We can conclude from the axioms of the linear structure that the identity map is not zero. Thus, the one-dimensional space $\mathrm{End}\,(a)$ for a sector a is spanned by the identity, and any symmetric map $f : a \to a$ from a sector $a \in \mathcal{S}$ to itself must be a multiple $f = \phi(f)\mathrm{id}_a$ of the identity. This establishes a one-to-one correspondence between sector endomorphisms f and scalars $\phi(f)$. In particular, this allows us to understand objects such as the trace $\mathrm{Tr}\,(g) : I \to I$ as a scalar. We are not careful about the distinction and write e.g. $\mathrm{Tr}\,(g)$ even if we mean its corresponding scalar $\phi(\mathrm{Tr}\,(g))$. In fact, taking the trace allows us to identify the prefactor since $\mathrm{Tr}\,(f) = \phi(f)\mathrm{Tr}\,(\mathrm{id}_a) = \phi(f)d_a$, where we adopt the shorthand notation $d_a = \dim a$ for the quantum dimension of a sector.

Further, since $\mathrm{Hom}\,(a,\ b)$ is zero-dimensional for sectors $a \neq b$, any map in it must be zero. Therefore, any map $f : a \to b$ between sectors a, b fulfills

$$(f : a \to b) = \delta_{a,b} \frac{\mathrm{Tr}\,(f)}{d_a}\ \mathrm{id}_a. \tag{5.58}$$

where we employ the notation abuse (5.50) for the Kronecker delta. We can understand this as a generalization of Schur's lemma. This relation is what eventually allows us to store symmetric tensors using fewer free parameters than non-symmetric tensors. The roadmap we follow in section 5.3 is to decompose a general symmetric tensor into components $c \to d$ that map between sectors. Then, because of (5.58), we only need to store those components $c \to c$ between *matching* sectors, and we only need to store one scalar prefactor per component.

Concrete case: Group Representation

In the group case, we note that irreducible representations (irreps) are simple because of Schur's Lemma; see section A.1. If a representation is reducible, it is equivalent to a direct sum of multiple irreps. Thus, the corresponding symmetry space is a direct sum of multiple simple symmetry spaces, and in particular, not itself simple. As a result, the simple objects in the group case correspond exactly to irreps.

Further, because of (5.5), an isomorphism of symmetry spaces implies an equivalence of the respective group representations, such that the sectors are equivalence classes of group irreps. For finite groups and compact Lie groups, the number of equivalence classes, and thus the number of sectors, is indeed countable.

The monoidal unit is the one-dimensional space $I = \mathbb{C}$ with the trivial representation $U_I(g) = 1$, which is clearly irreducible. Choosing it as the representative of its equivalence class makes it a sector.

By Schur's lemma part 2, any equivariant linear map from an irrep to itself is a multiple of the identity, such that the respective endomorphism space is indeed one-dimensional.

By Schur's lemma part 1, any equivariant linear map between inequivalent irreps is zero, such that the respective Homspace is indeed zero-dimensional. This result relies on the underlying field, here $\mathbb{C}$, being algebraically closed.

The decomposition (5.57) is derived in section A.1.

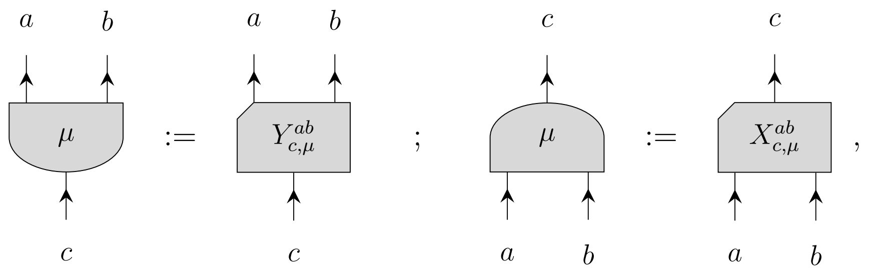
General case: Tensor Category

For a category, we simply take the listed properties as axioms; we assume there is a countable set $\mathcal{S} \subset \mathrm{Ob}(\mathbf{C})$ of objects, the *sectors* that fulfill the properties enumerated above.

Let us now pay particular attention to the sector decomposition

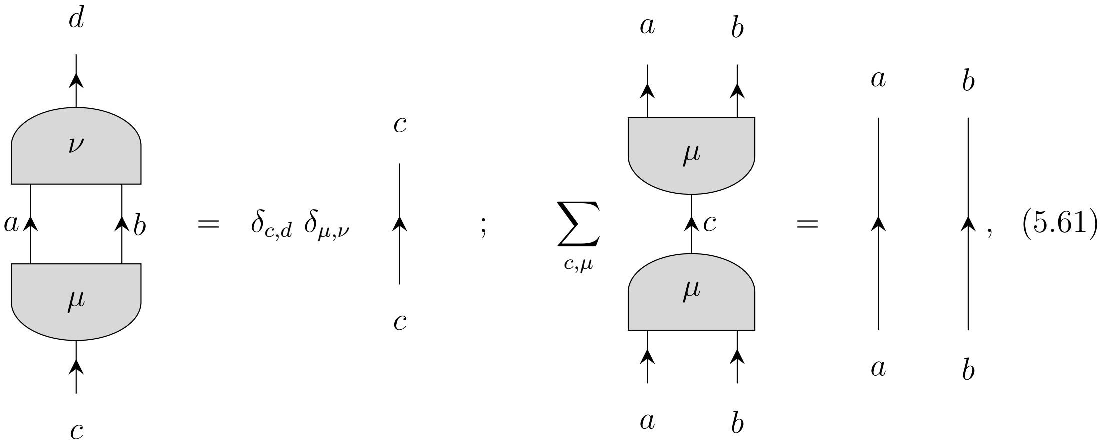
$$a \otimes b \cong \bigoplus_{c \in \mathcal{S}} \bigoplus_{\mu=1}^{N_c^{ab}} c \tag{5.59}$$

of the product of two sectors $a, b \in \mathcal{S}$. Here, the *N symbol* $N_c^{ab} := N_c^{a\otimes b}$, i.e. the number of times a sector c appears in the decomposition of $a \otimes b$, gets a special name and notation because it is used a lot. Similarly, the injections $Y^{ab}_{c,\mu} : c \to a \otimes b$ of the direct sum (5.59) play an important role and are called the *splitting tensors.* Here, we call $\mu = 1, \dots, N_c^{ab}$ the *multiplicity label.* The corresponding projections $X^{ab}_{c,\mu} = (Y^{ab}_{c,\mu})^\dagger : a \otimes b \to c$ are called *fusion tensors.* In the graphical notation, we introduce the following shorthand



(5.60)

where the sectors a, b, c are implied by the wires and only the multiplicity index μ is explicitly decorated. Note that the leg arrangement of the injections $Y^{ab}_{c,\mu}$ are visually intuitive, as they match the letter Y. Recall that as projections and inclusions, they fulfill orthonormality and completeness relations (5.52). In this case, they take the following explicit form.



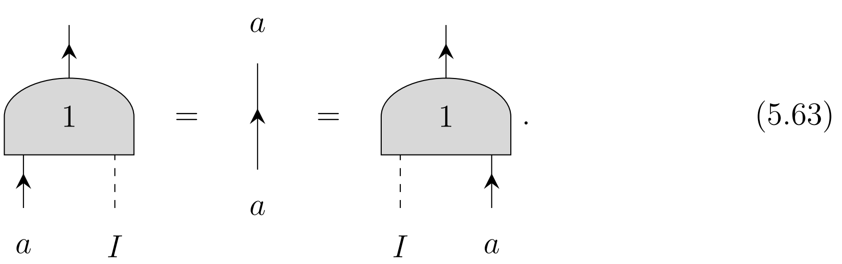
where the sum goes over all compatible fusion tensors, that is over $c \in \mathcal{S}$ and $\mu = 1, \ldots, N^{ab}_c$.

For fixed sectors a, b, c, we can view the $X^{ab}_{c,\mu}$ as an orthonormal basis for the space $\mathrm{Hom}\,(a \otimes b,\ c)$, indexed by μ. There is a gauge freedom in choosing this basis, and any unitary transformation

$$X^{ab}_{c,\mu} \mapsto \sum_{\nu} U_{\mu,\nu} X^{ab}_{c,\nu} \tag{5.62}$$

yields another set of valid fusion tensors. We propose to fix this gauge in section 5.2.2 to make the R symbol diagonal.

For fusion with the trivial sector, we have $a \otimes I \cong a \cong I \otimes a$ for any sector a, i.e. the sector decomposition only has one component a, and as a consequence we have $N^{aI}_b = \delta_{a,b} = N^{Ia}_b$. The only fusion tensor for the respective decompositions are given by the unitor isomorphisms of the monoidal structure, that is $X^{aI}_{a,1} = \rho_a$ and $X^{Ia}_{a,1} = \lambda_a$. Graphically, this reads



Note that we may restrict the first direct sum in (5.59) to the *fusion outcomes* of a and b that is to the set $\mathcal{F}_{a,b} := \{c \in \mathcal{S} | N_c^{a,b} > 0\}$ of unique sectors that actually appear.

Note that with the orthonormality relation (5.61), we have made a normalization choice. This normalization is natural in the case of a group symmetry since the fusion tensors are then given by the Clebsch-Gordan coefficients of the group representations. Another common choice is the isotopic normalization

$$(X_{\text{iso}})^{ab}_{c,\mu} := \left(\frac{d_a d_b}{d_c}\right)^{1/4} X^{ab}_{c,\mu} \tag{5.64}$$

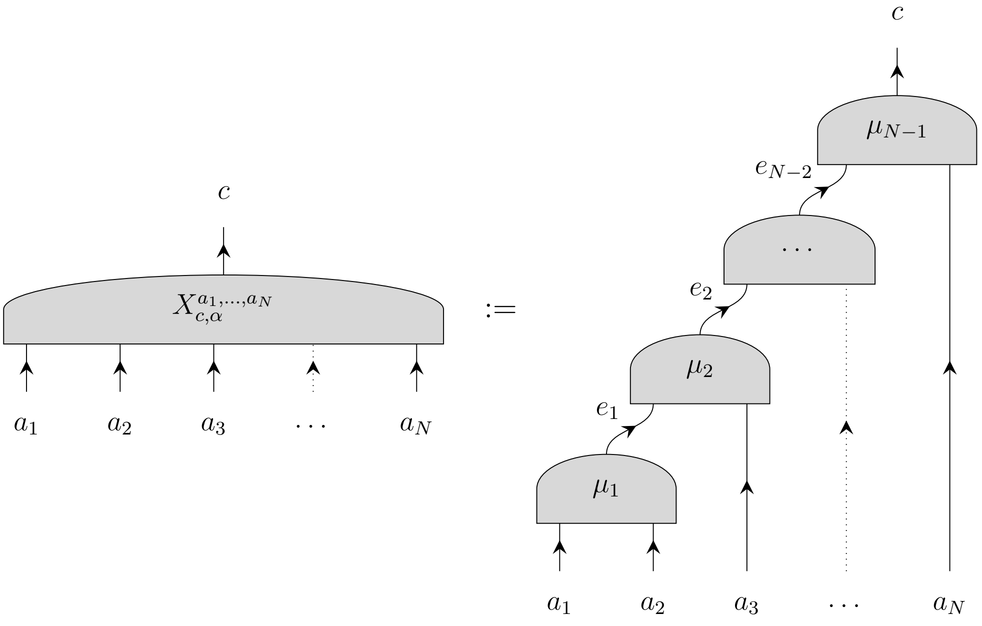
such that the analog of the orthonormality equation reads

$$(X_{\text{iso}})^{ab}_{c,\mu} \circ (Y_{\text{iso}})^{ab}_{c,\mu} = \sqrt{\frac{d_a d_b}{d_c}}\ \text{id}_c\ . \tag{5.65}$$

This choice simplifies the prefactors in (5.84), such that fusion tensors may consistently be drawn as vertices of wires, with no box.

5.1.9 Fusion Trees

We have seen the fusion and splitting tensors arise as projections and inclusions of the decomposition of the product of two sectors $a \otimes b$. The product $a_1 \otimes \cdots \otimes a_N$ of an arbitrary number N of sectors also decomposes as the direct sum of sectors. It turns out that we can construct projections (inclusions) of that direct sum from the fusion (splitting) tensors by arranging them in a tree structure, which we call a fusion (splitting) tree. In particular, consider the *fusion tree* $X^{a_1,\dots,a_N}_{c,\alpha} : a_1 \otimes \cdots \otimes a_N \to c$



(5.66)

which is labeled by a multi-index $\alpha = (e_1, \dots, e_{N-1}, \mu_1, \dots, \mu_{N-2})$. We call the $e_i \in \mathcal{S}$ the *inner sectors* and the $\mu_i \in \mathbb{N}$ the *inner multiplicities* of a tree. Taking the dagger gives us a *splitting tree* $Y_{c,\alpha}^{a_1,\dots,a_N} = (X_{c,\alpha}^{a_1,\dots,a_N})^\dagger : c \to a_1 \otimes \cdots \otimes a_N$, which graphically is just a mirrored fusion tree. Note that we have chosen a convention regarding the order of pairwise fusion; we always fuse the left-most pair of sectors first. We call the $a_1, \dots, a_N$ the *uncoupled* sectors of the tree and c the *coupled sector.*

A tree index α is *valid*, for fixed uncoupled and coupled sectors, if all of its inner multiplicities are within the correct ranges, meaning $1 \leq \mu_1 \leq N_{e_1}^{a_1 a_2}$, as well as $1 \leq \mu_n \leq N_{e_n}^{e_{n-1},a_{n+1}}$ for all $n = 2, \dots, N-2$, and $1 \leq \mu_{N-1} \leq N_c^{e_{N-2},a_N}$. There is one condition of consistent fusion per vertex of the tree. Note that this implies conditions on the inner sectors, namely that $e_1 \in \mathcal{F}_{a_1,a_2}$, and $e_n \in \mathcal{F}_{e_{n-1},a_{n+1}}$, as well as $c \in \mathcal{F}_{e_{N-2},a_N}$, since otherwise the respective N symbol is zero, such that no valid μ_n is possible.

The trees inherit orthonormality and completeness relations from the fusion and splitting tensors;

$$Y_{c,\alpha}^{a_1,\dots,a_N} \circ X_{d,\beta}^{a_1,\dots,a_N} = \delta_{c,d}\delta_{\alpha,\beta}\mathrm{id}_c \tag{5.67}$$

$$\sum_{c,\alpha} X_{c,\alpha}^{a_1,\dots,a_N} \circ Y_{c,\alpha}^{a_1,\dots,a_N} = \mathrm{id}_{a_1} \otimes \cdots \otimes \mathrm{id}_{a_N} \ , \tag{5.68}$$

where the sum goes over all valid trees α. We again understand the Kronecker delta in the sense of (5.50). This can easily be checked graphically by applying (5.61) $N-1$ times.

5.1.10 Terminology and Jargon

We assume a tensor category for our backend, which has all of the structures listed in the previous sections and all of the compatibility conditions between them. In the following, we list keywords for all of these structures as they may appear in the broader literature. For brevity, we do not list the various optional compatibility conditions between the separate structures.

balanced A braided monoidal category is balanced if it has a natural isomorphism $\theta_A : A \to A$, the twist which fulfills (5.43). We made the specific choice (5.42) for the balanced structure, which fulfills that property by construction.

biproducts A category has biproducts if for any finite set of objects $A_1, \dots, A_N$, the direct sum $\bigoplus_n A_n$ exists as defined in section 5.1.7.

braided A monoidal category is braided if it has a braiding structure as described in section 5.1.5.

dagger A dagger category is a category equipped with a dagger functor as described in section 5.1.3.

duals A category "has duals" if it is rigid.

enriched See linear. A linear category is also called enriched or **Vect**-enriched since its Homspaces can be understood as objects in the category **Vect** of vector spaces.

linear A category is ($\mathbb{C}$-)linear, if its homsets $\mathrm{Hom}\,(A,\ B)$ form ($\mathbb{C}$) vector spaces, with compatibility constraints, see section 5.1.6. Weaker assumptions exist, such as defining scalars to be morphisms $I \to I$, which form only a commutative semiring, not a field, and only requiring an addition rule, as e.g. done in reference [166, chpt. 2].

monoidal A category is monoidal if it has a tensor product as described in section 5.1.2.

pivotal A monoidal category with (right) duals is pivotal, if it has a monoidal natural transformation $\pi_A : A \to A^{\star\star}$, which can be shown to be an isomorphism. We made the specific choice (5.28) for a pivotal structure. Pivotality also guarantees the existence of left duals.

pre-fusion A pre-fusion category is a semisimple linear monoidal category.

ribbon (tortile) A ribbon category, a.k.a. a tortile category, is a balanced monoidal category with duals for which either of the following equivalent conditions hold; either the twist fulfills $(\theta_A)^{\mathrm{T}} = \theta_{A^\star}$ or equation (5.45) holds.

rigid A monoidal category is rigid if it has a (right) dual object $A^\star$ for every object A, with cup and cap maps η_A and ε_A, which fulfill the snake equation (5.21). This is a special case of the stronger structure we define in section 5.1.4, where we have directly used the dagger and defined a suitable pivotal structure in such a way that the chosen duals are both-sided dagger duals.

semisimple This notion is not as well established in the literature as the others. In a linear category with direct sums (a.k.a. biproducts), an object is simple if its endomorphism space is one-dimensional. It is semisimple if it is isomorphic to a finite direct sum of simple objects. The category is semisimple if all of its objects are semisimple. See section 5.1.8.

spherical A pivotal category is spherical if the left and right trace coincide, that is if the last equality in (5.32) holds.

tensor A tensor category, as we define it here, is a category that has all of the structures listed above, meaning it is a pivotal spherical pre-fusion ribbon dagger category, such that all the compatibility conditions hold. Note that slightly different – commonly weaker – definitions for the same term exist in the literature.

The following properties are not necessarily fulfilled by a tensor category but are, in principle, compatible, and there are relevant examples of tensor categories with these properties/structures.

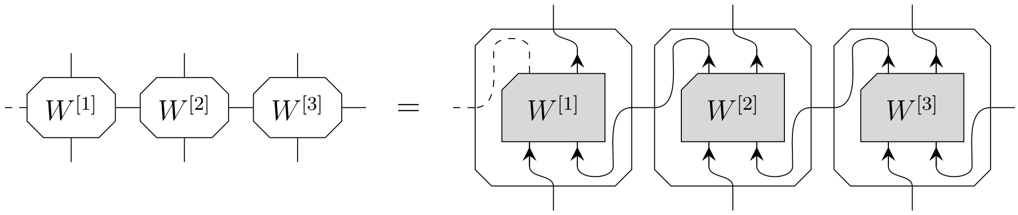
symmetric A symmetric category is a braided category where the braid is symmetric in the sense that $(\tau_{V,W})^{-1} = \tau_{W,V}$. This holds, e.g. for, group symmetries represented by the category $\mathbf{FdRep}_G$ or fermionic grading, represented by the category **Ferm**.

fusion category A fusion category is a pre-fusion category that is rigid and has a finite number of sectors. Thus, tensor categories are almost fusion categories, except that we allow a countably infinite number of sectors for tensor categories. As a non-example (a tensor category, which is not a fusion category), consider the category $\mathbf{FdRep}_{\mathrm{SU}(2)}$ of representations of SU(2), which has an infinite number of sectors, see section A.4.

modular A modular fusion category is a fusion category for which the modular S matrix (5.99) is invertible. They provide the mathematical framework for the theory of anyons and topological excitations. Note that for symmetric braiding, a fusion category can only be modular if there is only a single sector, which makes it trivial. In particular, this means that neither fermions nor the representations of a non-trivial group give rise to a modular fusion category. However, some examples, such as **Fib** described in section A.6, are modular and describe anyonic excitations.

5.1.11 Relation to graphical language of tensor networks

The graphical language for morphisms in a tensor category is closely related to, but slightly at odds with, the usual graphical notation for tensor networks, as employed, e.g., in chapter 2. Both approaches feature tensors as shapes in the plane and their legs/spaces as lines or wires. The largest difference is the additional meaning assigned to the positions of the endpoints of wires on a tensor at its bottom (top) for legs in the domain (codomain). We can easily reconcile this by assigning the legs of a tensor to either the domain or codomain and understand the tensor network notation as a lax version of the categorical notation, that allows moving the endpoints around if convenient. As an example, consider the following (part of a larger) MPO, expressed in the tensor network notation on the LHS and in the categorical language on the right.



(5.69)

In this fashion, the two pictures can seamlessly be identified for planar tensor networks. For non-planar diagrams, arising e.g. in the context of PEPS, see (2.59), the chirality of the braid induced by wire crossings needs to be specified explicitly, unless the symmetry has symmetric braiding. Note that arrows on the wires have a meaning in the categorical language and should not be used for other purposes, such that e.g. the arrow notation in [71], which describes isometric properties, is incompatible.

5.2 Topological data of a symmetry

In this section, we introduce and define several pieces of data – commonly referred to as the *topological data* of the symmetry – that are needed to do computations in practice. Note that we use "symmetry" as a broad term here, not limited to group symmetries, but referring to any tensor category, such as e.g. modular fusion categories describing anyons. We give concrete values (or explicit formulae) for some common symmetries in appendix A. We give a summary of the topological data, which can be used as an implementation guide in subsection 5.2.6.

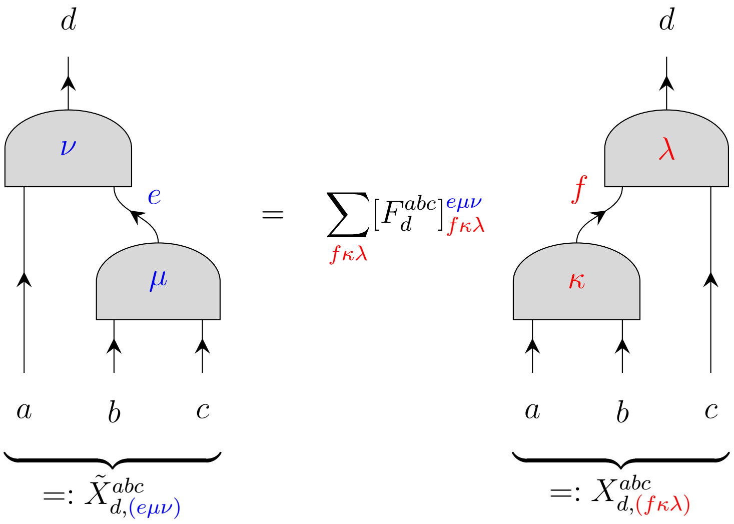
A recurring pattern in the following subsections is that we define a unitary symbol *implicitly*. We are able to do this because the symbol relates two different ONBs for a Homspace. For example, the F symbol is implicitly defined in equation (5.70). The diagrams on both sides form orthonormal bases of Hom $(a \otimes b \otimes c,\ d)$, indexed by the respective indices e, μ, ν (f, κ, λ). Thus, they are related by a unitary basis transformation[3]. This defines the F symbol as the matrix elements of that basis transformation. For the F symbol only, we also give an explicit definition of these matrix elements in equation (5.74). Similar explicit definitions can be written down for all other symbols as well but are omitted as they are not particularly insightful, and the form of a basis transformation is more useful.

5.2.1 Recoupling and F symbol

The F symbol is related to *recoupling* of fusion trees. We could have also chosen a different order of pairwise fusion to build the fusion trees. Such non-standard or non-canonical trees can always be written as linear combinations of standard trees since those are complete. The coefficients of these linear combinations give rise to

[3] Alternatively, unitarity can be proven explicitly. As a sketch: start from the orthonormality of one of the ONBs. Apply the symbol in both halves, where it is conjugated in the daggered half. Use the orthonormality of the other ONB to get back to an identity on the coupled sector. This equates the symbol, contracted with its matrix dagger, to a Kronecker delta, establishing unitarity.

the F symbol.


d
ν
e
μ
a b c
$=: \tilde{X}^{abc}_{d,(e\mu\nu)}$
$= \sum_{f\kappa\lambda} [F^{abc}_d]^{e\mu\nu}_{f\kappa\lambda}$
d
λ
f
κ
a b c
$=: X^{abc}_{d,(f\kappa\lambda)}$

(5.70)

For fixed sectors a, b, c, d, the F symbol is unitary as a matrix of the outer indices, that is

$$\begin{aligned} \sum_{e\mu\nu} [F^{abc}_d]^{e\mu\nu}_{f\kappa\lambda} [\overline{F}^{abc}_d]^{e\mu\nu}_{f'\kappa'\lambda'} &= \delta_{f,f'} \delta_{\kappa,\kappa'} \delta_{\lambda,\lambda'} \\ \sum_{f\kappa\lambda} [F^{abc}_d]^{e\mu\nu}_{f\kappa\lambda} [\overline{F}^{abc}_d]^{e'\mu'\nu'}_{f\kappa\lambda} &= \delta_{e,e'} \delta_{\mu,\mu'} \delta_{\nu,\nu'} . \end{aligned} \tag{5.71}$$

Since F is unitary, the reverse transformation reads

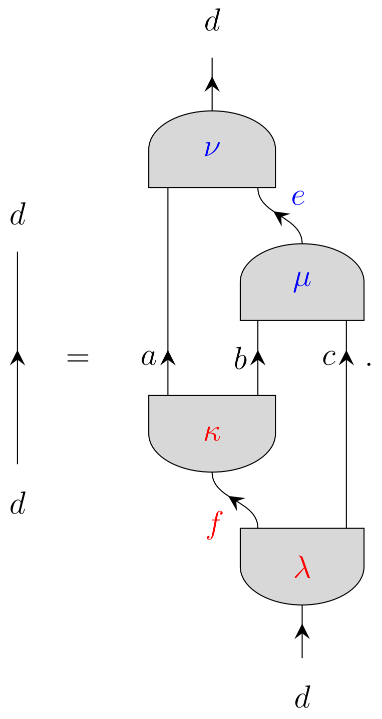
$$X^{abc}_{d,(f\kappa\lambda)} = \sum_{e\mu\nu} [\overline{F}^{abc}_d]^{e\mu\nu}_{f\kappa\lambda} \tilde{X}^{abc}_{d,(e\mu\nu)}, \tag{5.72}$$

which would admit a nice graphical representation similar to (5.70), but is omitted here for brevity. Taking the dagger gives us two more relations, describing the recoupling of non-canonical splitting trees $\tilde{Y} := \tilde{X}^\dagger$ to canonical splitting trees $Y = X^\dagger$.

$$\tilde{Y}^{abc}_{d,(e\mu\nu)} = \sum_{f\kappa\lambda} [\overline{F}^{abc}_d]^{e\mu\nu}_{f\kappa\lambda} Y^{abc}_{d,(f\kappa\lambda)} \quad ; \quad Y^{abc}_{d,(f\kappa\lambda)} = \sum_{e\mu\nu} [F^{abc}_d]^{e\mu\nu}_{f\kappa\lambda} \tilde{Y}^{abc}_{d,(e\mu\nu)} \tag{5.73}$$

In addition to the implicit definition above, let us include an explicit form of the F symbol. To obtain it from equation (5.70), we right-compose with $(X^{abc}_{d,(f'\kappa'\lambda')})^\dagger$ and use orthonormality (5.67) to collapse the sum and find

$[F^{abc}_d]^{e\mu\nu}_{f\kappa\lambda}$



(5.74)

To get back to the implicit definition (5.70), use completeness (5.68) of the canonical (red) trees. Alternatively, use analogous completeness relations of the non-canonical (blue) trees, to obtain equation (5.72).

The F symbol fulfills its own pentagon consistency equation

$$\sum_{h\gamma\delta\epsilon}[F^{abc}_i]^{h\gamma\epsilon}_{j\lambda\rho}[F^{ahd}_e]^{g\delta\kappa}_{i\epsilon\omega}[F^{bcd}_g]^{f\mu\nu}_{h\gamma\delta} = \sum_{\sigma}[F^{jcd}_e]^{g\nu\kappa}_{j\lambda\sigma}[F^{jcd}_e]^{f\mu\sigma}_{i\rho\omega}\ , \tag{5.75}$$

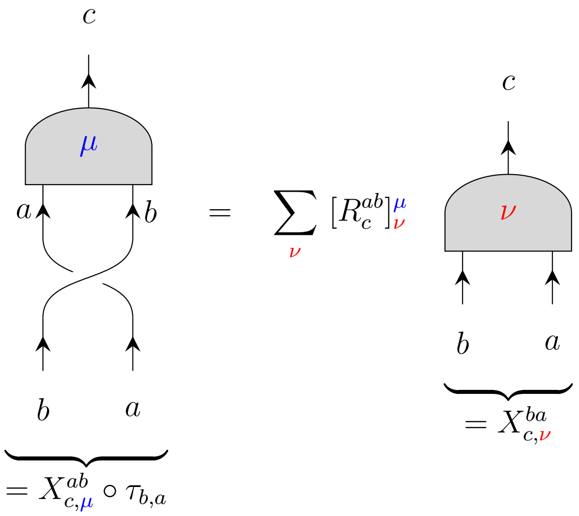
which we can view either as inherited from the pentagon equation (5.17) of the associator[4]. Alternatively, we can equate the coefficients that arise in the two inequivalent ways of recoupling a 4-to-1 fusion tree $a \otimes (b \otimes (c \otimes d)) \to e$ to a canonical tree $((a \otimes b) \otimes c) \otimes d \to e$.

5.2.2 Braiding and R symbol

Next up, we introduce the topological data related to braids: the R symbol. We can think of the R symbol as the matrix elements of the braid $\tau_{a,b}$ in the basis given by fusion / splitting tensors. Alternatively, think about braiding the legs below a fusion tensor $a \otimes b \to c$. Since the braid is unitary, this composite object still gives an orthonormal and complete set, parametrized by the multiplicity index of the fusion tensor above the braid. The R symbol is the unitary basis transformation

[4] We can view the F symbol as matrix elements of the associator α in a basis given by the fusion (splitting) tensors. The graphical notation in equation (5.74) implies an associator $\alpha_{a,b,c}$ in the middle of the diagram, to compose $\mathrm{id}_a \otimes X^{bc}_{e,\mu}$ with $Y^{ab}_{f,\kappa} \otimes \mathrm{id}_c$.

that relates it to the standard fusion tensors $b \otimes a \to c$. It is implicitly defined as the following coefficients.


c
μ
a
b
$=$
$\sum_{\nu} [R^{ab}_c]^{\mu}_{\nu}$
c
ν
b
a
b
a
$= X^{ab}_{c,\mu} \circ \tau_{b,a}$
$= X^{ba}_{c,\nu}$

(5.76)

Note the convention for the order of upper indices. We find this order practical since we use the R symbol when we have a fusion tensor with uncoupled sectors a, b and apply a braid to it. We can use the gauge freedom (5.62) of the fusion tensors such that the R symbol is diagonal $[R^{ab}_c]^{\mu}_{\nu} \propto \delta_{\mu,\nu}$.

We obtain a relation for applying an under-braid below a fusion tensor from unitarity of the R symbol

$$X^{ab}_{c,\nu} \circ \tau^{\dagger}_{a,b} = \sum_{\mu} [\overline{R^{ba}_c}]^{\mu}_{\nu} X^{ba}_{c,\mu} \ . \tag{5.77}$$

Taking the dagger gives us relations for braiding above splitting tensors

$$\tau^{\dagger}_{b,a} \circ Y^{ab}_{c,\mu} = \sum_{\nu} [\overline{R^{ab}_c}]^{\mu}_{\nu} Y^{ba}_{c,\nu} \qquad ; \qquad \tau_{a,b} \circ Y^{ab}_{c,\nu} = \sum_{\mu} [R^{ba}_c]^{\mu}_{\nu} Y^{ba}_{c,\mu} \ . \tag{5.78}$$

Note that we get a permutation $a \leftrightarrow b$ on the upper indices for an underbraid on a fusion tensor X and for an overbraid on a splitting tensor Y.

The twist (5.42) is directly related to braiding. By (5.58), the twist $\theta_a \in \mathrm{End}\,(a)$ of a sector a must be a multiple of the identity, which implicitly defines the prefactor $\Theta_a \in \mathbb{C}$ such that

$$\theta_a = \Theta_a \mathrm{id}_a \ . \tag{5.79}$$

Since the literature is inconsistent on whether θ_a or Θ_a is "the twist", we will not carefully distinguish them either. Since the twist is unitary, we have $\Theta_a \overline{\Theta_a} = 1$, meaning Θ_a is a complex phase. The twist is contained[5] in the R symbol as

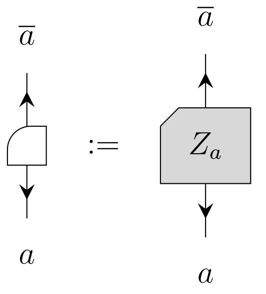
$$\Theta_a = \sum_{b \in \mathcal{S}} \sum_{\mu=1}^{N^{aa}_b} \frac{d_b}{d_a} [R^{aa}_b]^{\mu}_{\mu}. \tag{5.80}$$

[5] A sketch of the derivation; Insert a resolution (5.61) of identity above the braid in the definition (5.42). Then, use (5.58) and (5.33).

5.2.3 Dual sectors and Z isomorphism

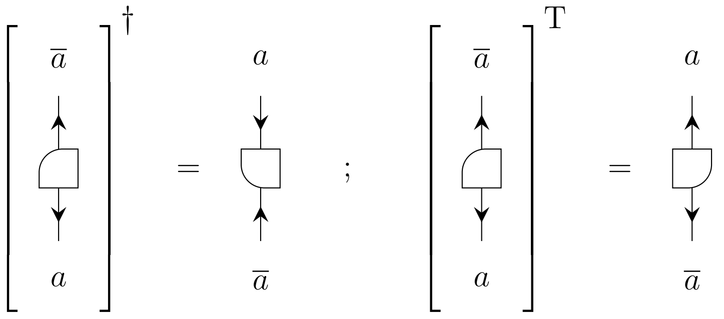
Let us now consider the dual space $a^\star$ of a sector a. We know it is a simple space since $f \mapsto f^{\mathrm{T}}$ is a vector space isomorphism, establishing $\mathrm{End}\,(a) \cong \mathrm{End}\,(a^\star)$ and thus that $\mathrm{End}\,(a^\star)$ is one-dimensional. The dual space $a^\star$ is, however (in general) not a sector since it may not be the representative of its isomorphism class[6]. We write $\overline{a} \in \mathcal{S}$ for the sector that is isomorphic to $a^\star \cong \overline{a}$ and call it the *dual sector* (of a). Since $\overline{a}^\star \cong a^{\star\star} \cong a$, we have $\overline{\overline{a}} = a$, i.e. a is the dual sector of $\overline{a}$.

We write $Z_a : a^\star \overset{\cong}{\rightarrow} \overline{a}$ for the isomorphism. We may choose it to be unitary by rescaling. Graphically, we represent it by an unlabelled smaller box with a rounded (instead of chamfered) corner. The sector a is clear from the wire below.



(5.81)

Let us state the graphical notation for the dagger and for the transpose at this point, as they might not be entirely intuitive.

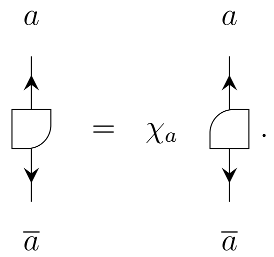


(5.82)

Recall that the dagger of a diagram is given by a mirror *plus flipping arrow directions back*. We also observe that both $(Z_a)^{\mathrm{T}}$ and $Z_{\overline{a}}$ are non-zero elements of $\mathrm{Hom}\,(\overline{a}^\star,\ a)$. Since that Homspace between simple objects is one-dimensional, they must be proportional. This defines the *Frobenius Schur indicator* χ_a as the prefactor

[6] In some cases, *but not in general*, this can be gauged away by redefining the representatives. Assume for every sector a we have either $a^\star = a$ or $a^\star \ncong a$. Then, we can redefine the sectors, that is, redefine which simple object represents each isomorphism class. In the first case, $a^\star$ already is a sector, and in the second case, we may define $a^\star$ as the representative for its class, i.e. as a sector. However, if there is a sector a such that $a \cong a^\star \neq a$, i.e. such that a and $a^\star$ are in the same sector but not equal, the representative is already fixed to a and we can not have $a^\star$ as a representative at the same time.

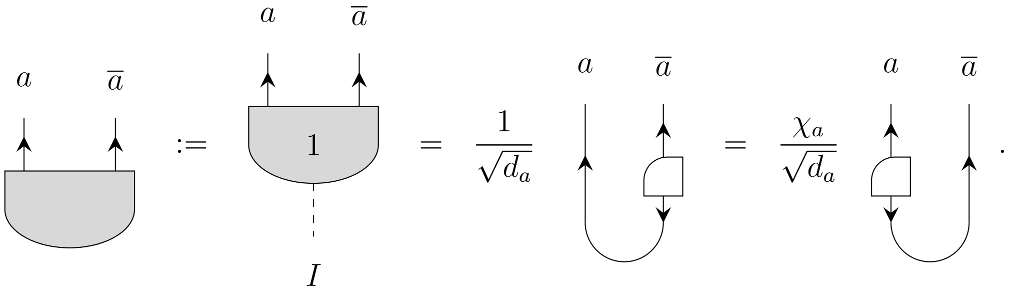
$Z_a^{\mathrm{T}} = \chi_a Z_{\overline{a}}$. Graphically, this reads



(5.83)

The Frobenius Schur indicator is constrained[7] by $\chi_a = \chi_{\overline{a}} = \pm 1$.

Further, we can choose[8] the phase of the Z isomorphism such that the splitting tensor $Y^{a\overline{a}}_{I,1}$ is related to the cups by



(5.84)

Taking the dagger relates the fusion tensor $X^{a\overline{a}}_{I,1}$ to the caps.

We can plug these relations into the definition (5.74) of the F symbol to find that

$$[F^{a\overline{a}a}_{a}]^{I11}_{I11} = \frac{\chi_a}{d_a}. \tag{5.85}$$

Since $d_a > 0$ and $\chi_a = \pm 1$ we obtain the following relations

$$\chi_a = \operatorname{sgn}[F^{a\overline{a}a}_{a}]^{I11}_{I11} \tag{5.86}$$

$$d_a = \frac{1}{|[F^{a\overline{a}a}_{a}]^{I11}_{I11}|} \tag{5.87}$$

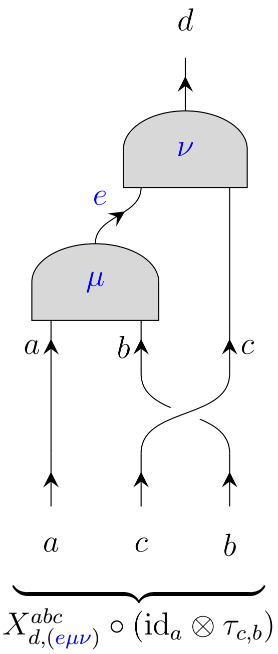
which tells us that (and how) both the Frobenius Schur indicator and quantum dimension are contained in the F symbol and are not independent data.

[7] Unitarity of Z_a implies $|\chi_a|^2 = 1$, i.e. χ_a is a complex phase. Taking the transpose of (5.83), and then applying it again we find $1 = \chi_a \chi_{\overline{a}}$ and therefore $\chi_{\overline{a}} = \overline{\chi_a}$. Now, if $a \neq \overline{a}$, we can redefine the representative for the sector $\overline{a}$ such that $\chi_a = 1$. For $a = \overline{a}$, we have $1 = \chi_a \chi_{\overline{a}} = \chi_a^2$, that is $\chi_a = \pm 1$. In either case $\chi_{\overline{a}} = \overline{\chi_a} = \chi_a$.

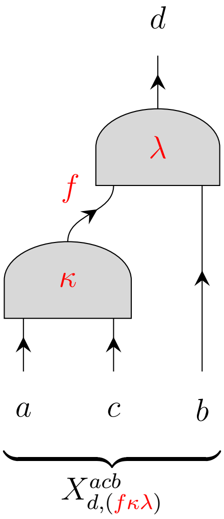
[8] Let us sketch the derivation: Consider applying first $(Z_a)^\dagger$ and then a cap to the right wire in (5.84). Both operations are reversible such that the resulting equation is equivalent. Note the LHS is a map $a \to a$ and thus a multiple of the identity. To derive/confirm the magnitude of the prefactor, compose (5.84) with its dagger. The phase of the prefactor can be chosen by redefining $Z_a \mapsto \mathrm{e}^{\mathrm{i}\phi} Z_a$, fixing the phase of the Z isomorphism relative to the phase of the fusion tensors.

5.2.4 Braiding fusion trees and C symbol

While the R symbol defined in section 5.2.2 contains all relevant data for braiding, the following composite symbol is also useful in practice. It relates a (part of a larger) fusion tree with a braid to canonical fusion trees. The C symbol is defined as the coefficients in the following equation.


d
ν
e
μ
a
b
c
a
c
b
$X^{abc}_{d,(e\mu\nu)} \circ (\mathrm{id}_a \otimes \tau_{c,b})$

$$= \sum_{f\kappa\lambda} [C^{abc}_d]^{e\mu\nu}_{f\kappa\lambda} \qquad (5.88)$$



As for the R symbol, we choose the order of upper indices abc to match the fusion tree $X^{abc}_{d,(e\mu\nu)}$ on the LHS before it is braided.

The analogous relation for an underbraid follows from unitarity

$$X^{abc}_{d,(f\kappa\lambda)} \circ (\mathrm{id}_a \otimes \tau^\dagger_{b,c}) = \sum_{e\mu\nu} [\overline{C}^{acb}_d]^{e\mu\nu}_{f\kappa\lambda} X^{acb}_{d,(e\mu\nu)}. \qquad (5.89)$$

Taking the dagger gives us relations for braiding above splitting trees, namely

$$(\mathrm{id}_a \otimes \tau_{b,c}) \circ Y^{abc}_{d,(f\kappa\lambda)} = \sum_{e\mu\nu} [C^{acb}_d]^{e\mu\nu}_{f\kappa\lambda} Y^{acb}_{d,(e\mu\nu)} \qquad (5.90)$$

$$(\mathrm{id}_a \otimes \tau^\dagger_{c,b}) \circ Y^{abc}_{d,(e\mu\nu)} = \sum_{f\kappa\lambda} [\overline{C}^{abc}_d]^{e\mu\nu}_{f\kappa\lambda} Y^{acb}_{d,(f\kappa\lambda)}. \qquad (5.91)$$

Similar to the R move relations, we see a permutation $b \leftrightarrow c$ in the upper indices of the C symbol for the relations with an underbraid $\tau^\dagger$ on a fusion tree X or an overbraid τ on a splitting tree Y.

We can naively derive an expression for the C symbol by acting on the LHS as follows; First, we do an inverse F move to get a fusion tensor above the braid, resolve the braid with an R move, and then get back to the canonical structure with a forward F move. As a result we find

$$[C^{abc}_d]^{e\mu\nu}_{f\kappa\lambda} = \sum_{g\alpha\beta\gamma} [\overline{F}^{abc}_d]^{g\alpha\beta}_{e\mu\nu} [R^{bc}_g]^{\alpha}_{\gamma} [F^{abc}_d]^{g\gamma\beta}_{f\kappa\lambda}. \quad \text{(inefficient, use (5.93) in practice!)} \tag{5.92}$$

We can, however, obtain a more practical expression by using coherence to "lift" the c wire over the bottom fusion tensor, resolving the resulting underbraid of a, c with an inverse R move, recoupling to the canonical tree structure with a single F move, and finally resolving the braid of e, c with another R move. We obtain

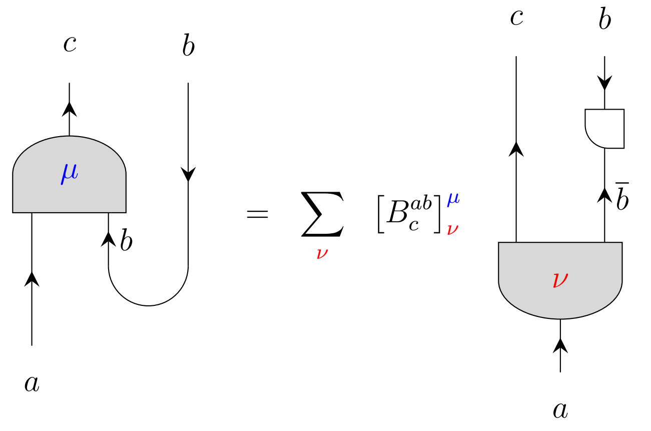
$$[C^{abc}_d]^{e\mu\nu}_{f\kappa\lambda} = \sum_{\alpha\beta} [R^{ec}_d]^{\nu}_{\alpha} [F^{cab}_d]^{e\mu\alpha}_{f\beta\lambda} [\overline{R}^{ac}_f]^{\kappa}_{\beta} \tag{5.93}$$

which is cheaper to use in practice since there is only one expensive F symbol and there is no sum over sectors.

5.2.5 Bending lines and B symbol

Another operation that we need to do to fusion tensors in practice is "bending lines", that is e.g. applying a cup below a fusion tensor. It turns out that due to our choice of fusion trees, we only ever need to bend the right leg of a fusion tensor.

We implicitly define the B symbol as the coefficients in the following linear combination.



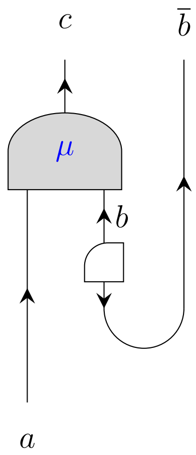
(5.94)

Note that we need to include a Z isomorphism to get the correct arrow directions. Note that while both sides are orthogonal complete sets, only the composites on the RHS are normalized. Therefore, the B symbol is *not* unitary. It is instead normalized[9] such that

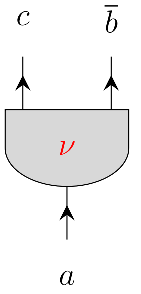
$$\sum_{\nu} [B^{ab}_c]^{\mu}_{\nu} [\overline{B}^{ab}_c]^{\mu'}_{\nu} = \frac{d_c}{d_a}\delta_{\mu,\mu'} \quad ; \quad \sum_{\mu} [B^{ab}_c]^{\mu}_{\nu} [\overline{B}^{ab}_c]^{\mu}_{\nu'} = \frac{d_c}{d_a}\delta_{\nu,\nu'} \; . \tag{5.95}$$

[9] To derive the normalization, compose (5.94) with its dagger, use (5.58), cyclic property of the trace and orthonormality of fusion tensors.

If there was already a Z isomorphism present before bending the line, we can first slide it over using (5.31), then after the regular B move use (5.83) to flip either one of the Z isomorphisms, such that it cancels the other.



$$= \sum_{\nu} \chi_b \left[B^{ab}_c\right]^{\mu}_{\nu} \qquad (5.96)$$



Again, relations for bending lines on a splitting tensor can be obtained by taking the dagger.

The B symbol can be obtained from the F symbol as follows. Start with the LHS of (5.94), use (5.84) on the cup, insert $(\rho_a)^\dagger = Y^{aI}_{a,1}$ below and recouple the splitting tree with an F move to find

$$[B^{ab}_c]^{\mu}_{\nu} = \sqrt{d_b}[\overline{F}^{ab\bar{b}}_a]^{I11}_{c\mu\nu}. \qquad (5.97)$$

5.2.6 Summary of topological data

In an implementation of a tensor backend, the following data/functions of a symmetry are strictly needed.

(i) A data format for sector labels. Preferably, this should be a simple and hashable data structure, such that fusion tree indices are hashable. We propose (arrays of) integers, e.g. by storing $2S \in \mathbb{Z}$, twice the half-integer quantum numbers of SU(2) irreps.

(ii) The N symbol (5.59) and F symbols (5.70), which encode the fusion of sectors.

(iii) The R symbol (5.76), which encodes braiding.

(iv) If the number of sectors is infinite, a function to enumerate the possible fusion outcomes $\mathcal{F}_{a,b} := \{c \in \mathcal{S} | N^{a,b}_c > 0\}$ is required, to obtain finite loops. It is convenient even if the number of sectors is finite.

The following data can be obtained from the data above. These relations can be used as fallback (default) implementations. For a concrete symmetry, however, it is often possible to simplify the expressions, allowing more efficient implementations.

(v) The C symbol, see (5.88) with fallback implementation (5.93).

(vi) The B symbol, see (5.94) with fallback implementation (5.97).

(vii) The quantum dimension d_a, see (5.34), with fallback implementation (5.87).

(viii) The Frobenius Schur indicator χ_a, see (5.83), with fallback (5.86).

(ix) The twist Θ_a, see (5.79), with fallback implementation (5.80).

For a group symmetry (but not in general!), symmetric maps are linear maps between vector spaces. We can thus convert symmetric tensors to (or from) their explicit matrix elements, that is, to (from) a representation that does not enforce the symmetry. In order to do this, explicit matrix representations of the following maps are required

(x) The fusion and splitting tensors (5.60). They are given by the Clebsch-Gordan coefficients.

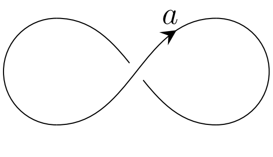
(xi) The Z isomorphism (5.81). They may be a bit tricky to work out in practice and are rarely tabulated. Note that the prefactor is determined by unitarity and by demanding that equation (5.84) holds. For one-dimensional sectors, this fully determines Z isomorphism. This is enough for abelian groups, where all sectors are one-dimensional. For Lie groups, the concrete derivation that we give for SU(2) in section A.4.1 should readily generalize.

This concludes the necessary and optional data associated with a symmetry.

5.2.7 Anyon data

Now let us assume that the tensor category is a modular fusion category, meaning in addition to the properties we require of a tensor category, it additionally has a finite set of sectors, making it a fusion category and has an invertible S matrix (5.99), making it modular. These modular fusion categories describe quantum states of anyonic excitations, and in that context, the following quantities are common and may serve as a sanity check to make sure that the category does indeed describe the anyon theory it is supposed to describe.

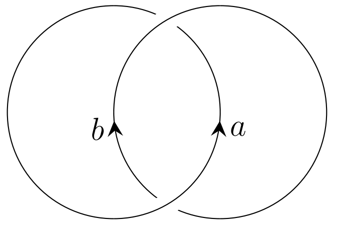
The modular T matrix is the following composite

$$T_{a,b} := \delta_{a,b}$$


$$= \delta_{a,b}\mathrm{Tr}\left(\theta_a\right) = \delta_{a,b}\Theta_a d_a. \qquad (5.98)$$

The modular S matrix is the following composite

$$S_{a,b} := \frac{1}{\mathcal{D}}$$



$$= \frac{1}{\mathcal{D}} \sum_{c\in\mathcal{S}} \sum_{\mu,\nu=1}^{N_c^{ab}} d_c \big[R_c^{ba}\big]^{\mu}_{\nu} \big[R_c^{ab}\big]^{\mu}_{\nu}, \tag{5.99}$$

where $\mathcal{D} := \sqrt{\sum_a d_a^2}$ is the total quantum dimension of the theory. The expression in terms of R symbols can be derived by inserting a resolution of identity (5.61) in terms of fusion tensors above the braids and using two R moves (5.76), then using that the trace is cyclic and finally orthonormality of fusion tensors.

5.3 Symmetric tensors

In this section, we establish the free parameters of symmetric tensors, as well as how to do common operations, such as leg rearrangement, contraction, or decomposition on them.

Let us first introduce some terminology. A symmetric tensor is a symmetric map

$$T : W_1 \otimes \cdots \otimes W_K \to V_1 \otimes \cdots \otimes V_J \tag{5.100}$$

from the *domain* $\mathcal{W} := W_1 \otimes \cdots \otimes W_K$ to the *codomain* $\mathcal{V} := V_1 \otimes \cdots \otimes V_J$. If there are no spaces in the (co-)domain, that is, if $K = 0$ ($J = 0$), we understand the empty tensor product to mean the monoidal unit $\mathcal{W} = I$ ($\mathcal{V} = I$). By bending a bunch of lines, i.e. applying cups below, the tensor T is equivalent to a map

$$\tilde{T} : I \to V_1 \otimes \cdots \otimes V_J \otimes (W_K)^\star \otimes \cdots \otimes (W_1)^\star. \tag{5.101}$$

Thus we define the *legs* of a tensor T to be $V_1, \ldots, V_J, (W_K)^\star, \ldots, (W_1)^\star$, i.e. is the spaces in the codomain, followed by the duals of the spaces in the domain *in reverse order*, see figure 5.1. As a result, the legs do not change if we bend lines, which allows a tensor backend to hide the bipartition of legs into codomain and domain from high-level functionality as an implementation detail.

It is convenient in practice to keep track of duality explicitly, that is, to allow any number of legs on the tensor to have an explicit duality star and to understand a general tensor as, e.g. a map

$$T : W_1 \otimes (W_2)^\star \otimes \cdots \otimes W_K \to (V_1)^\star \otimes \cdots \otimes V_J, \tag{5.102}$$

where we picked some example positions for the explicit duality stars, but any of the spaces may or may not be explicitly dual.

The strategy is to identify components of a tensor that are maps between sectors, such that (5.58) applies. That is, we want to systematically build the sector decomposition of the entire domain (codomain). We want to do this in such a way that

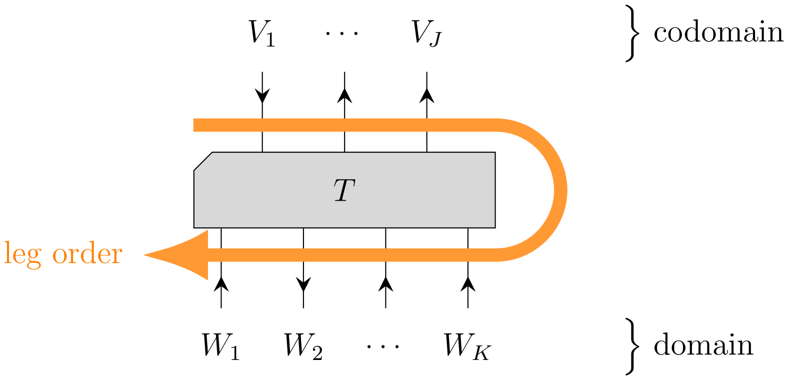


Figure 5.1: We choose a canonical order of legs as the legs of the codomain, followed by the duals of the legs in the domain in reverse order.

each leg of the tensor is treated on an equal footing so that we can readily permute (braid) the legs or move them between codomain and domain. Therefore, we start from the sector decomposition (5.57), of each individual leg, e.g.

$$V_k \cong \bigoplus_a \bigoplus_{n=1}^{N_a^{V_k}} a, \tag{5.103}$$

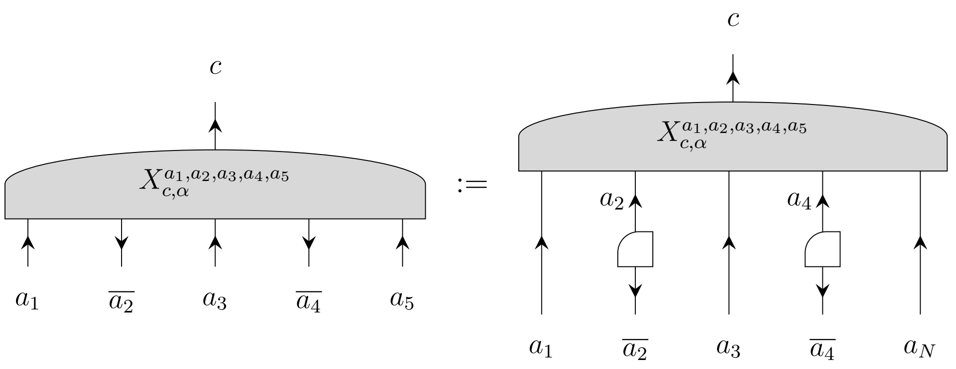
with projections $p_{a,n}^{V_k}$ and inclusions $i_{a,n}^{V_k} = (p_{a,n}^{V_k})^\dagger$. If we had an explicitly dual space instead, we find

$$(V_k)^\star \cong \bigoplus_a \bigoplus_{n=1}^{N_a^{V_k}} a^\star \tag{5.104}$$

since we can construct projections $p_{a^\star,n}^{(V_k)^\star} := (i_{a,n}^{V_k})^\mathrm{T}$ and inclusions as their daggers. This allows us to extract components $a_1 \otimes (a_2)^\star \otimes \cdots \otimes a_K \to (b_1)^\star \otimes \cdots \otimes b_J$ that map between tensor products of (duals of) sectors. Unlike the abelian case discussed in section 2.5, we can not formulate a charge rule at this level. Instead, we need to extract components that map from a single sector to another single sector.

5.3.1 Generalized Fusion Trees

This basis change can be provided by the fusion trees (5.66), if we allow the following modification. The fusion trees only allow sectors as inputs, not their duals. To also capture duals, we can include Z isomorphisms where appropriate and define the following composite as a (generalized) fusion tree.



(5.105)

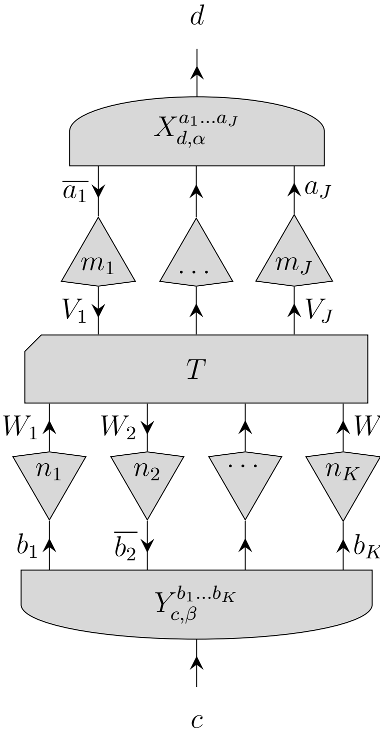
Note that we still label the generalized fusion tree with the sectors above the layer of possible Z isomorphisms, not by its input sectors. This is chosen because these sectors are relevant for the behavior of the fusion tree under manipulations, such as braids. The presence of Z isomorphisms is unambiguously implied by the arrow directions.

A (generalized) fusion tree is thus fully specified by the following data.

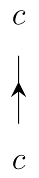
- The uncoupled sectors $a_1, \dots, a_N$.
- The coupled sector c.
- The inner sectors $e_1, \dots, e_{N-2}$.
- The multiplicity labels $\mu_1, \dots, \mu_{N-1}$.
- A boolean flag for every uncoupled sector, indicating if there is a Z isomorphism below or not.

5.3.2 Parametrization of symmetric tensors

Now, we can insert a bunch of resolutions of identity both above and below the tensor T, first projecting each leg to a single sector via the projections of (5.103) or (5.104), then mapping the uncoupled sectors to a single coupled sector with a generalized fusion tree (5.105). The resulting components are maps $c \to d$ between sectors



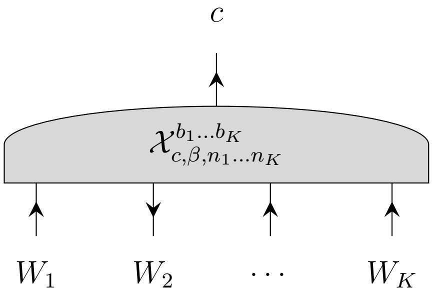
$$\overset{(5.58)}{=:} \quad \delta_{c,d} \quad \Big[\big[T_c\big]^{a_1\ldots a_J,\alpha}_{b_1\ldots b_K,\beta}\Big]^{m_1\ldots m_J}_{n_1\ldots n_k} \tag{5.106}$$



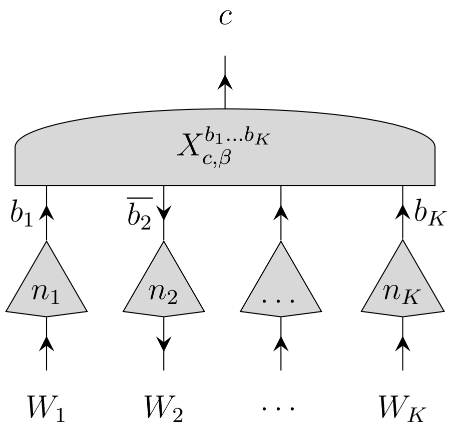
and thus are multiples of the identity $c \to c$ if $c = d$ and zero otherwise. This defines the free parameters $[[T_c]^{a_1\ldots a_J,\alpha}_{b_1\ldots b_K,\beta}]^{m_1\ldots m_J}_{n_1\ldots n_k}$ of a symmetric tensor as the respective prefactors.

Note that for the explicitly dual spaces – $(W_2)^\star$ and $(V_1)^\star$ in this example – we label the components and fusion trees by the sectors b_2 and a_1 of the original spaces W_2 and V_1, even though $\overline{b_2}$ and $\overline{a_1}$ appear in the LHS diagram.

The composite projections



$$:= \tag{5.107}$$



and the related inclusions $\mathcal{Y} = \mathcal{X}^\dagger$ inherit orthonormality

$$\mathcal{X}^{b_1\ldots b_K}_{c,\beta,n_1\ldots n_K} \circ \mathcal{Y}^{a_1\ldots a_K}_{d,\alpha,m_1\ldots m_K} = \delta_{a_1,b_1}\ldots\delta_{a_K,b_K}\ \delta_{m_1,n_1}\ldots\delta_{m_K,n_K}\ \delta_{c,d}\ \delta_{\alpha,\beta}\ \mathrm{id}_c \tag{5.108}$$

and completeness

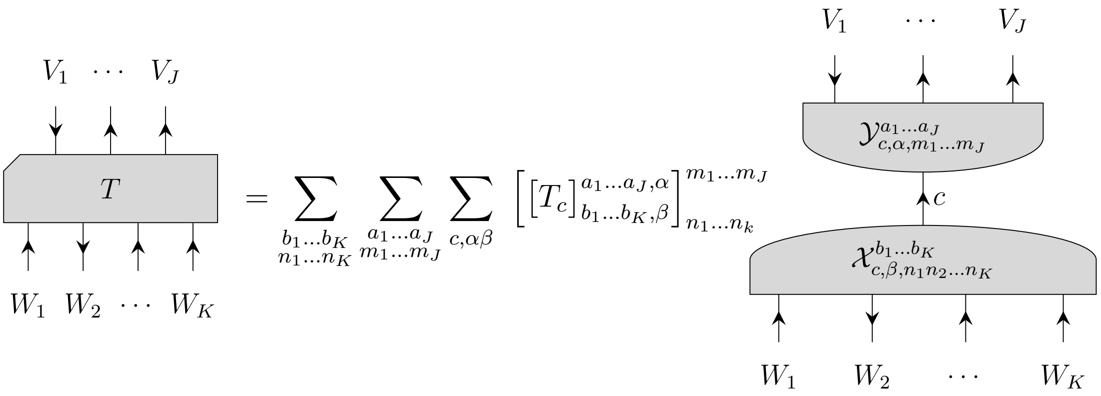
$$\sum_{b_1\ldots b_K}\sum_{m_1\ldots m_K}\sum_{c,\beta}\mathcal{Y}^{b_1\ldots b_K}_{c,\beta,m_1\ldots m_K}\circ\mathcal{X}^{b_1\ldots b_K}_{c,\beta,m_1\ldots m_K}=\mathrm{id}_{W_1}\otimes\mathrm{id}_{W_2^\star}\otimes\cdots\otimes\mathrm{id}_{W_K} \tag{5.109}$$

relations.

Thus, we can compute the components as

$$\left[[T_c]^{a_1\ldots a_J,\alpha}_{b_1\ldots b_K,\beta}\right]^{m_1\ldots m_J}_{n_1\ldots n_k}\mathrm{id}_c=\mathcal{X}^{a_1\ldots a_J}_{c,\alpha,m_1\ldots m_J}\circ T\circ\mathcal{Y}^{b_1\ldots b_K}_{c,\beta,n_1\ldots n_K}\ . \tag{5.110}$$

As a consequence of completeness, the free parameters are enough to describe any tensor since we can reconstruct the original tensor as


V_1 $\cdots$ V_J
T
W_1 W_2 $\cdots$ W_K
$=\sum_{\substack{b_1\ldots b_K\\ n_1\ldots n_K}}\sum_{\substack{a_1\ldots a_J\\ m_1\ldots m_J}}\sum_{c,\alpha\beta}\left[[T_c]^{a_1\ldots a_J,\alpha}_{b_1\ldots b_K,\beta}\right]^{m_1\ldots m_J}_{n_1\ldots n_k}$
V_1 $\cdots$ V_J
$\mathcal{Y}^{a_1\ldots a_J}_{c,\alpha,m_1\ldots m_J}$
c
$\mathcal{X}^{b_1\ldots b_K}_{c,\beta,n_1n_2\ldots n_K}$
W_1 W_2 $\cdots$ W_K

(5.111)

5.3.3 Blocks

We can group those prefactors $[[T_c]^{a_1\ldots a_J,\alpha}_{b_1\ldots b_K,\beta}]^{m_1\ldots m_J}_{n_1\ldots n_K}$ that belong to a given pair of fusion and splitting tree into a *tree block* $[T_c]^{a_1\ldots a_J,\alpha}_{b_1\ldots b_K,\beta}$. The tree block is a $\mathcal{T}^{\mathcal{V}}_{a_1\ldots a_J}\times\mathcal{T}^{\mathcal{W}}_{b_1\ldots b_K}$, matrix, where the *tree block sizes* are given by

$$\mathcal{T}^{\mathcal{V}}_{a_1\ldots a_J}:=\prod_{j=1}^{J}N^{V_j}_{a_j}\ . \qquad \text{(tree block size)} \tag{5.112}$$

Note that the width (height) of a tree block $[T_c]^{a_1\ldots a_J,\alpha}_{b_1\ldots b_K,\beta}$ depends only on its upper (lower) indices. We can thus stack the tree blocks to form larger matrices.

There is an intermediate level in the hierarchy that we call a *forest block* $[T_c]^{a_1\ldots a_J}_{b_1\ldots b_K}$ which comprises all the tree blocks with the same coupled sector c and uncoupled sectors $a_1\ldots a_J$ and $b_1\ldots b_K$. They are $\mathcal{F}^{\mathcal{V}}_{a_1\ldots a_J,c}\times\mathcal{F}^{\mathcal{W}}_{b_1\ldots b_K,c}$ matrices, where the size is given by

$$\mathcal{F}^{\mathcal{V}}_{a_1\ldots a_J,c}=\mathcal{T}^{\mathcal{V}}_{a_1\ldots a_J}N^{a_1\ldots a_J}_c\ . \qquad \text{(forest block size)} \tag{5.113}$$

Here, $N_c^{a_1 \dots a_J}$ is the number of valid fusion trees $a_1 \otimes \dots \otimes a_J \to c$, which we can recursively define as

$$N_c^{a_1 \dots a_J} = \sum_{f \in \mathcal{S}} N_c^{f, a_J} N_f^{a_1 \dots a_{J-1}} \tag{5.114}$$

with the N symbol as the base case for $J = 2$, $N_c^a = \delta_{a,c}$ for $J = 1$ and $N_c^I = \delta_{c,I}$ for $J = 0$.

Finally, we can stack all those forest blocks with the same coupled sector c (or directly all the tree blocks) to a *block* T_c. It is a $\mathcal{B}_c^{\mathcal{V}} \times \mathcal{B}_c^{\mathcal{W}}$ matrix, where the *block sizes* are given by

$$\mathcal{B}_c^{\mathcal{V}} = \sum_{a_1 \dots a_J} \mathcal{F}^{\mathcal{V}}_{a_1 \dots a_J, c} = \sum_{a_1 \dots a_J} \mathcal{T}^{\mathcal{V}}_{a_1 \dots a_J} N_c^{a_1 \dots a_J}. \qquad \text{(block size)} \tag{5.115}$$

This turns out to be the multiplicity of the sector c in the domain $\mathcal{V}$, i.e.

$$\mathcal{V} = V_1 \otimes \dots \otimes V_J \cong \bigoplus_c \bigoplus_{\mu=1}^{\mathcal{B}_c^{\mathcal{V}}} c \; . \tag{5.116}$$

We visualize the structure of a block in figure 5.2.

The reason to organize the data in this particular way is that the blocks are the largest collection of free parameters that allows the most expensive tensor operations – contraction and decomposition – to directly go through to the block level, as we discuss in the following subsections. However, permuting the legs (e.g. braiding them or bending lines) requires breaking the blocks apart to either the forest or tree level and recombining them according to coefficients that arise from manipulating the trees.

Since there might be an infinite number of sectors, we can not save the blocks T_c "for all" c. Instead, we store only the nonzero blocks together with the coupled sectors c to which they correspond.

5.3.4 Basic tensor operations

Let us first discuss some basic operations on tensors. As a pattern for this and the following subsections, we consider some operation (e.g. addition) of symmetric tensors and ask how we can build the blocks of the output tensor from the blocks of the input tensor(s). Generally, they are derived explicitly from equations (5.110) and (5.111).

Let us first consider linear combinations. For tensors $F, G : \mathcal{V} \to \mathcal{W}$ and scalars $a, b \in \mathbb{C}$, we find that linear combinations simply go through to the block level, that is

$$(aF + bG)_c = aF_c + bG_c. \tag{5.117}$$

This is because the expression for the free parameters (5.110) is linear in the tensor.

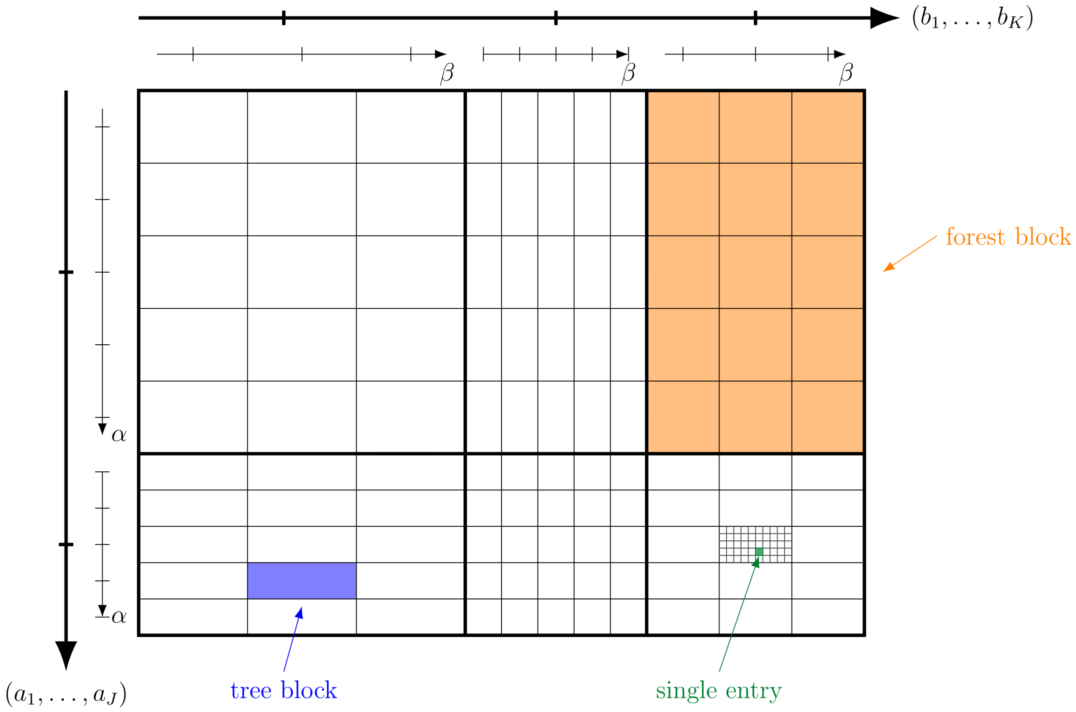


Figure 5.2: Illustration of a block T_c. Blocks are matrices that contain all free parameters for a fixed coupled sector c. It consists of forest blocks (one example in orange), which contain the entries for fixed uncoupled sectors $(a_1,\dots,a_J)$ and $(b_1,\dots,b_K)$. Those are further partitioned into the tree blocks (one example in blue) $[T_c]^{a_1\dots a_J,\alpha}_{b_1\dots b_K,\beta}$, which additionally fixes a fusion tree α and a splitting tree β. Those tree blocks contain the scalar entries $[[T_c]^{a_1\dots a_J,\alpha}_{b_1\dots b_K,\beta}]^{m_1\dots m_J}_{n_1\dots n_K}$ (one example in green).

The identity map $\mathrm{id}_\mathcal{V}$ as a tensor $\mathcal{V}\to\mathcal{V}$ can be built using identity blocks

$$(\mathrm{id}_\mathcal{V})_c = \mathbb{1}_{\mathcal{B}^\mathcal{V}_c\times\mathcal{B}^\mathcal{V}_c}\ . \tag{5.118}$$

Taking the dagger of a tensor $T:\mathcal{V}\to\mathcal{W}$ yields a tensor $T^\dagger:\mathcal{W}\to\mathcal{V}$ whose blocks

$$(T^\dagger)_c = (T_c)^\dagger \tag{5.119}$$

are given by the blockwise matrix dagger.

5.3.5 Combining and splitting legs

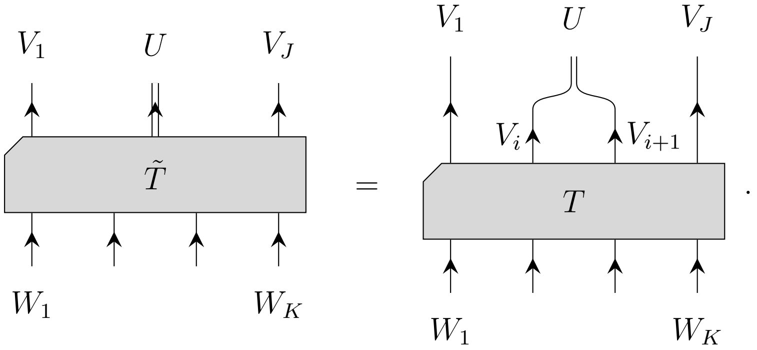
To combine legs, we want to derive the block structure of a tensor

$$\tilde{T} : W_1 \otimes \cdots \otimes W_K \to V_1 \otimes \cdots \otimes V_{i-1} \otimes U \otimes V_{i+2} \otimes \cdots \otimes V_J \tag{5.120}$$

that arises from

$$T : W_1 \otimes \cdots \otimes W_K \to V_1 \otimes \cdots \otimes V_i \otimes V_{i+1} \otimes \cdots \otimes V_J \tag{5.121}$$

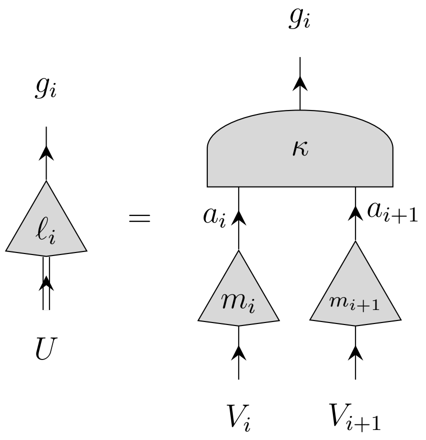
by combing the legs $U := V_i \otimes V_{i+1}$. That is graphically



(5.122)

Note that even though the right-most upper leg has index J, the tensor $\tilde{T}$ only has $J-1$ upper legs. In the following, we derive how the blocks (free parameters) of $\tilde{T}$ are related to those of T, which facilitates combining or splitting legs.

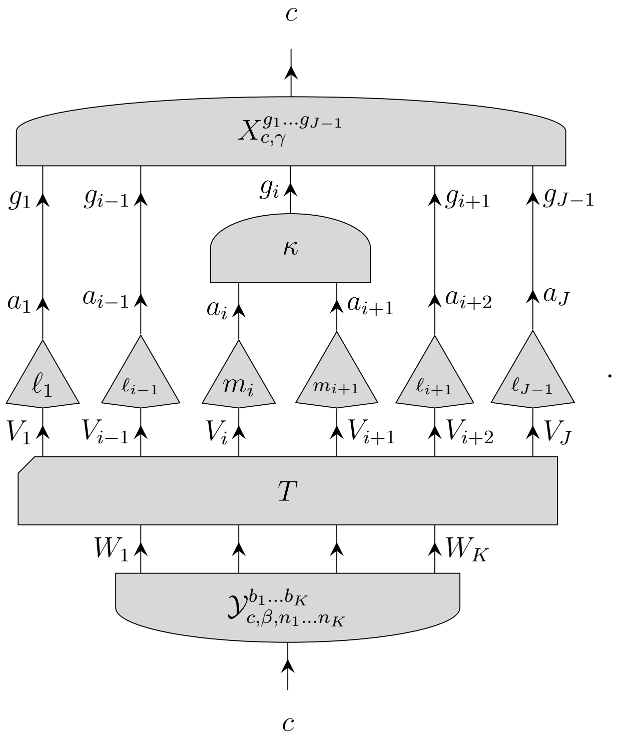
As a first step, we can build the sector projections of the product space $U = V_i \otimes V_{i+1}$ as



(5.123)

with the multi-index $\ell_i := (a_i, m_i, a_{i+1}, m_{i+1}, \kappa)$. We obtain for the free parameters of $\tilde{T}$

$$\left[\left[\tilde{T}_c\right]^{g_1\ldots g_{J-1},\gamma}_{b_1\ldots b_K,\beta}\right]^{\ell_1\ldots\ell_{J-1}}_{n_1\ldots\ldots n_K} \quad \overset{c}{\underset{c}{\uparrow}} \quad =$$



.

(5.124)

Now, we can use (5.70) to recouple the single fusion tensor into the fusion tree and find

$$\begin{aligned}&\left[\left[\tilde{T}_c\right]^{g_1\ldots g_{J-1},(e_1\ldots e_{J-3},\mu_1\ldots\mu_{J-2})}_{b_1\ldots b_K,\beta}\right]^{\ell_1\ldots\ell_{J-1}}_{n_1\ldots\ldots n_K}\\ &\quad=\sum_{f\lambda\pi}\left[F^{e_{i-2},a_i,a_{i+1}}_{e_{i-1}}\right]^{g_i,\kappa,\mu_{i-1}}_{f\lambda\pi}\left[\left[T_c\right]^{a_1\ldots a_J,(e_1\ldots f\ldots,e_{J-3},\mu_1\ldots\lambda\pi\ldots\mu_{J-2})}_{b_1\ldots b_K,\beta}\right]^{m_1\ldots m_J}_{n_1\ldots\ldots n_K}\end{aligned} \tag{5.125}$$

where on the RHS, $(a_1 \ldots a_{i-1}) := (g_1 \ldots g_{i-1})$, while a_i, a_{i+1} are part of the multi-index $\ell_i = (a_i, m_i, a_{i+1}, m_{i+1}, \kappa)$ and $(a_{i+2} \ldots a_J) := (g_{i+1} \ldots g_{J-1})$ have an index shift. The indices ℓ_j and m_j behave analogously. On the RHS, the fusion tree index is modified such that f and λ are inserted at position $i-1$ respectively and π replaces μ_{i-1}, i.e. the fusion tree index in the RHS has inner sectors $(e_1 \ldots e_{i-2}, f, e_{i-1}, \ldots e_{J-3})$ and multiplicity labels $(\mu_1 \ldots \mu_{i-2}, \lambda, \pi, \mu_i, \ldots \mu_{J-2})$.

Strictly, the expression above is only valid for $2 < i < J-1$. For $i = 2$, we need to swap $e_{i-2} \mapsto a_1$ in the upper indices of the F symbol, such that this sector always refers to the sector to the bottom left of the node μ_{i-1} in the fusion tree. For $i = J-1$, we need to substitute $e_{i-1} \mapsto c$ in the lower index, such that this sector always refers to the sector above the node μ_{i-1} in the fusion tree.

In the special case $i = 1$, that is, combing two legs at the very left, we do not need to do an F move since we can directly understand the additional fusion tensor as the bottom-most node of a larger fusion tree and find

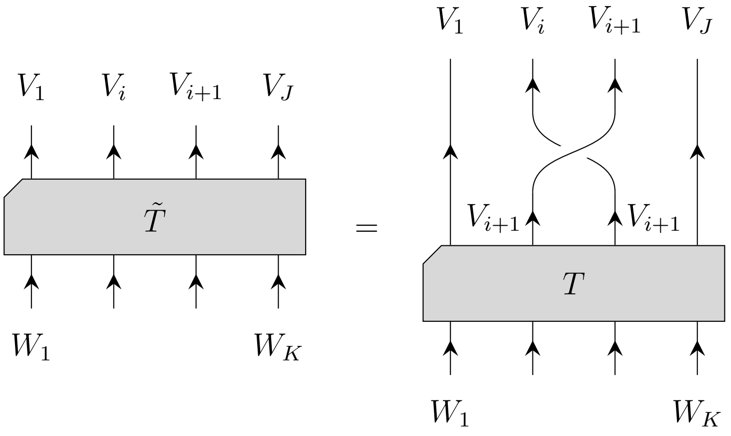
$$\left[\left[\tilde{T}_c\right]^{g_1 \ldots g_{J-1},(e_1 \ldots e_{J-3},\mu_1 \ldots \mu_{J-2})}_{b_1 \ldots b_K,\beta} \right]^{\ell_1 \ldots \ell_{J-1}}_{n_1 \ldots \ldots n_K} = \left[\left[T_c\right]^{a_1 \ldots a_J,(g_1,e_1 \ldots e_{J-3},\kappa\mu_1 \ldots \mu_{J-2})}_{b_1 \ldots b_K,\beta} \right]^{m_1 \ldots m_J}_{n_1 \ldots \ldots n_K} . \tag{5.126}$$

Combining legs in the domain is analogous, except that the index modifications happen in the subscripts instead of the superscripts and that the F symbol is complex conjugated, as in equation (5.73).

In order to do the reverse operation and split a leg, invert equation (5.125), using unitarity of the F symbol.

5.3.6 Braiding Legs

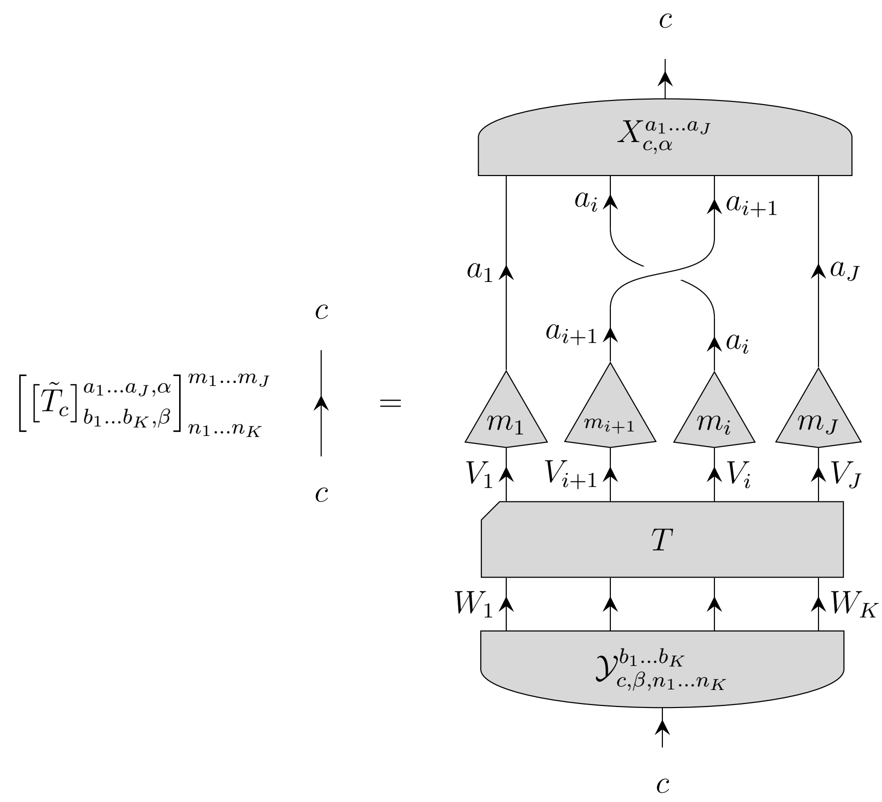
We discuss general rearrangement of legs in section 5.3.8. In preparation, let us first consider a single braid between neighboring legs in the codomain, that is a tensor $\tilde{T}$ that results from braiding the i-th leg over the $(i+1)$-th leg of a tensor T.



(5.127)

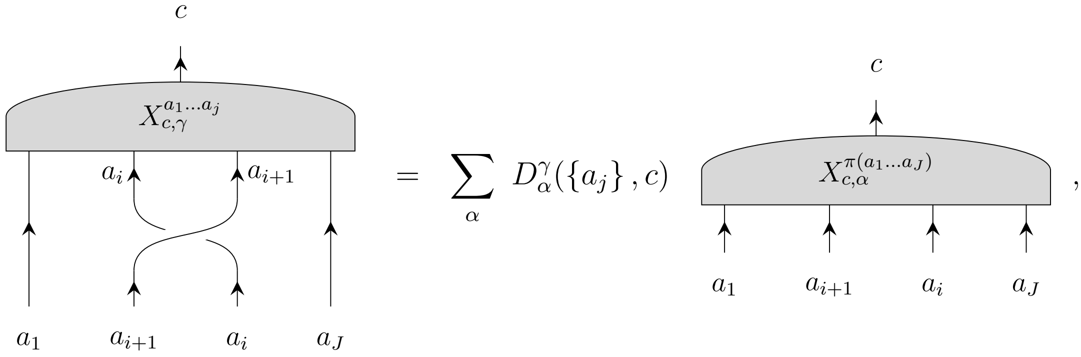
We employ a notation that, if parsed strictly, suggests $1 < i < J-1$, i.e. excludes braids at the very left or very right, and we depict graphically the specific case of $i = 2$ and $J = 4$. This allows us to simplify the notation. We understand this as a sketch of a general derivation, which proves the same results for $1 \leq i \leq J-1$ and any $J \geq 2$.

Let us write $\pi(V_1 \ldots V_J) = (V_1 \ldots V_{i+1} V_i \ldots V_J)$ for the permutation on the upper legs. We can now plug the RHS into the definition (5.106) and use the sliding property (5.39) to slide the projections below the braid.


$\left[\left[\tilde{T}_c\right]^{a_1\dots a_J,\alpha}_{b_1\dots b_K,\beta}\right]^{m_1\dots m_J}_{n_1\dots n_K}$
c
c
$=$
c
$X^{a_1\dots a_J}_{c,\alpha}$
a_i
a_{i+1}
a_1
a_J
a_{i+1}
a_i
m_1
m_{i+1}
m_i
m_J
V_1
V_{i+1}
V_i
V_J
T
W_1
W_K
$\mathcal{Y}^{b_1\dots b_K}_{c,\beta,n_1\dots n_K}$
c

(5.128)

This sliding step induces the same permutation π to also act on the indices m_j of the projections. It remains to deal with the braid acting below a fusion tree. To relate to the blocks of T, we need to write it as a linear combination of regular fusion trees


c
$X^{a_1\dots a_j}_{c,\gamma}$
a_i
a_{i+1}
a_1
a_{i+1}
a_i
a_J
$= \sum_\alpha D^\gamma_\alpha(\{a_j\}\,,c)$
c
$X^{\pi(a_1\dots a_J)}_{c,\alpha}$
a_1
a_{i+1}
a_i
a_J
,

(5.129)

such that

$$\left[\left[\tilde{T}_c\right]^{a_1\dots a_J,\gamma}_{b_1\dots b_K,\beta}\right]^{m_1\dots m_J}_{n_1\dots n_K} = \sum_\alpha D^\gamma_\alpha(\{a_j\}\,,c)\ \left[\left[T_c\right]^{\pi(a_1\dots a_J),\alpha}_{b_1\dots b_K,\beta}\right]^{\pi(m_1\dots m_J)}_{n_1\dots n_K}. \tag{5.130}$$

We can derive the coefficients D from an R move (5.76) if $i = 1$ or a C move (5.88) otherwise. These coefficients are highly sparse, i.e. they vanish for most α. Instead of polluting notation with many Kronecker deltas, let us list only those contributions that are (in general) nonzero. To that end, let $\gamma = (e_1 \dots e_{J-2}, \mu_1 \dots \mu_{J-1})$ be a general but fixed tree index.

For $i = 1$, assuming the gauge of fusion tensors is chosen such that the R symbols are diagonal, we only get a nonzero contribution if $\alpha = \gamma$. In that case, we have

$$D^{\gamma}_{\gamma}(\{a_j\}, c) = \left[R^{a_1,a_2}_{e_1}\right]^{\mu_1}_{\mu_1} . \qquad \text{(all other } D^{\gamma}_{\alpha} = 0) \tag{5.131}$$

For braids at $2 < i < J-1$, we find non-zero contribution only for those α that arise from γ by modifying the inner sector $e_{i-1} \mapsto f$ and multiplicity labels $(\mu_{i-1}, \mu_i) \mapsto (\kappa, \lambda)$. In that case, we have

$$\begin{gathered} D^{\gamma}_{(e_1 \dots f \dots e_{J-2}, \mu_1 \dots \kappa\lambda \dots \mu_{J-1})}(\{a_j\}, c) = \left[C^{e_{i-2},a_i,a_{i+1}}_{e_i}\right]^{e_i \mu_{i-1} \mu_i}_{f \kappa \lambda} \\ \text{(all other } D^{\gamma}_{\alpha} = 0) \; . \end{gathered} \tag{5.132}$$

The expression above generalizes to $i = 2$ if we understand $e_0 := a_1$, such that e_{i-1} always refers to the sector to the bottom left of the vertex μ_i. Similarly it generalizes to $i = J-1$ if we understand $e_{J-1} := c$, such that e_i is the sector above μ_i.

To perform an underbraid in the domain instead, we simply need to take the complex conjugate of the coefficients and exchange the roles of upper and lower indices in (5.130), that is let the permutation act on the $\{b_k\}$ and $\{n_k\}$ instead, and form the linear combination on the index β of splitting trees instead of α, with coefficients $\overline{D}^{\delta}_{\beta}(\{b_k\}, c)$.

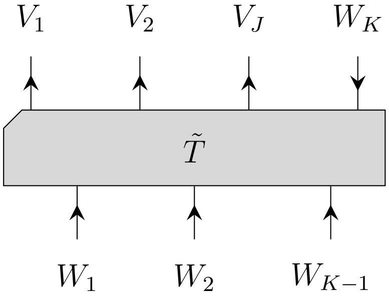
To swap the chirality of the braid, that is perform an underbraid in the codomain or an overbraid in the domain, we need the following modifications in addition to the above. Take the complex conjugate of the coefficients and swap $a_i \leftrightarrow a_{i+1}$ in the superscript of the R or C symbol, as discussed in section 5.2.4.

Braids can be composed such that an expression of the form of (5.130) can be derived for composites of many braids, as long as they act on only the codomain. Figuring out the coefficients D for the composite braid and evaluating (5.130) only once is more practical than evaluating the free parameters for all intermediate objects. The composition is easily done by composing the permutations π and forming the matrix product of the coefficient matrices D. Note that since the D coefficients typically show a high level of sparsity, it is advantageous to collect only those fusion trees for which there is a nonzero coefficient, i.e. to build a mapping (e.g. a python dictionary, or hashtable), assigning to each contributing fusion tree α its coefficient D^{γ}_{α} in (5.130), similar to algorithm 5.1.

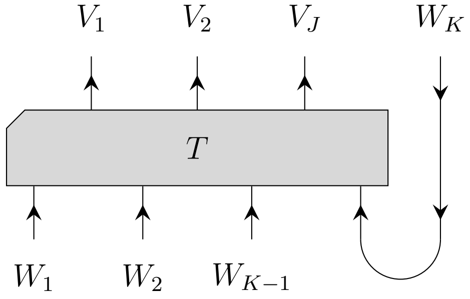
In practice, we can implement braiding in the codomain by acting row-wise on the blocks – recall the structure summarized in figure 5.2. That is, for every block we can perform the following steps to realize equation (5.130). First, within each “tree row”, that is for fixed $(c, a_1 \dots a_J, \alpha)$, permute the rows of single entries according to π. Second, within each “forest row”, that is for fixed $(c, a_1 \dots a_J)$, form linear combination of the tree rows according to the coefficients $D^{\gamma}_{\alpha}(\{a_j\}, c)$. Lastly, for the whole block, permute the resulting forest rows according to π. Braiding in the domain can be done similarly by acting on entire columns.

5.3.7 Bending Lines

Next, let us consider bending a single line at the very right of the domain

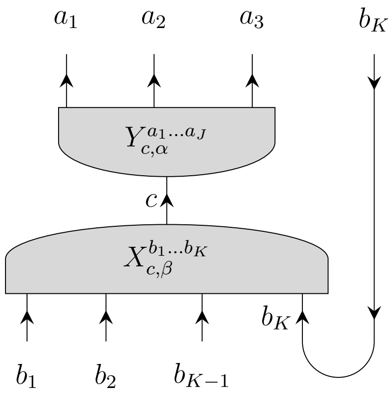


$$=$$

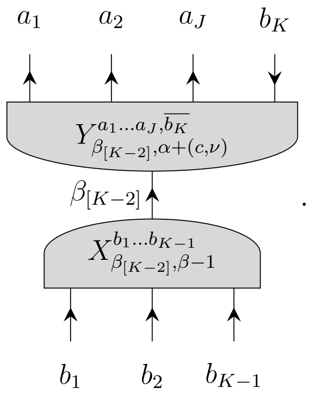


. (5.133)

To derive an expression for the blocks of $\tilde{T}$ in terms of the blocks of T, we plug in the parametrization (5.111) of T, then use the sliding property (5.31) to slide the projection across the bend. This cooperates with the structure (5.111) of free parameters, since the transpose of the projection $(p_{n_k} : W_k \to b_k)^{\mathrm{T}} : b_k^\star \to W_k^\star$ is the inclusion i_{n_k} of the dual space. We then need to consider how the bend acts on the pair of fusion and splitting tree. Let $\alpha = (\alpha_{[1]} \dots \alpha_{[J-2]}, \alpha^{(1)} \dots \alpha^{(J-1)})$ and similarly $\beta = (\beta_{[1]} \dots \beta_{[K-2]}, \beta^{(1)} \dots \beta^{(K-1)})$ denote tree indices. Then, we find from a B move (5.94) that



$$= \sum_\nu \left[B^{\beta_{[K-2]} b_K}_c\right]^{\beta^{(K-1)}}_\nu$$


a_1
a_2
a_J
b_K
$Y^{a_1\dots a_J,\overline{b_K}}_{\beta_{[K-2]},\alpha+(c,\nu)}$
$\beta_{[K-2]}$
$X^{b_1\dots b_{K-1}}_{\beta_{[K-2]},\beta-1}$
b_1
b_2
b_{K-1}

. (5.134)

Here we write $\alpha + (c, \nu) = (\alpha_{[1]} \dots \alpha_{[J-2]}, c, \alpha^{(1)} \dots \alpha^{(J-1)}, \nu)$ for an extended fusion tree index and $\beta - 1 = (\beta_{[1]} \dots \beta_{[K-3]}, \beta^{(1)} \dots \beta^{(K-2)})$ for a shortened index, with the last sector and last multiplicity label removed. Shifting around the sums in (5.111), we can identify the free parameters of the transformed tensor as

$$\left[\left[\tilde{T}_d\right]^{a_1\dots a_{J+1},\alpha+(c,\nu)}_{b_1\dots b_{K-1},\beta'}\right]^{m_1\dots m_{J+1}}_{n_1\dots n_{K-1}} = \sum_\mu \left[B^{d,\bar{a}_{J+1}}_c\right]^\mu_\nu \left[\left[T_c\right]^{a_1\dots a_J,\alpha}_{b_1\dots b_{K-1}a_{J+1},\beta'+(d,\mu)}\right]^{m_1\dots m_J}_{n_1\dots n_{K-1}m_{J+1}}, \tag{5.135}$$

where we relabelled the new coupled sector $\beta_{[K-2]} \mapsto d$ and a multiplicity index $\beta^{(K-1)} \mapsto \mu$, as well as $\bar{b}_K \mapsto a_{J+1}$ and $n_K \mapsto m_{J+1}$, since those indices now belong to the codomain of $\tilde{T}$.

Bending a leg from the codomain can be derived analogously, with the result

$$\left[\left[\tilde{T}_d\right]^{a_1\dots a_{J-1},\alpha'}_{b_1\dots b_{K+1},\beta+(c,\nu)}\right]^{m_1\dots m_{J-1}}_{n_1\dots n_{K+1}} = \sum_{\mu} \left[\overline{B}^{d,\bar{b}_{K+1}}_{c}\right]^{\mu}_{\nu} \left[\left[T_c\right]^{a_1\dots a_{J-1}b_{K+1},\alpha'+(d,\mu)}_{b_1\dots b_K,\beta}\right]^{m_1\dots m_{J-1}n_{K+1}}_{n_1\dots n_K}. \tag{5.136}$$

If the bent line was an explicitly dual space before bending it, the above expression obtains an additional factor $\chi_{b_K} = \chi_{a_{J+1}}$, since we need to use (5.95) instead, but is otherwise analogous.

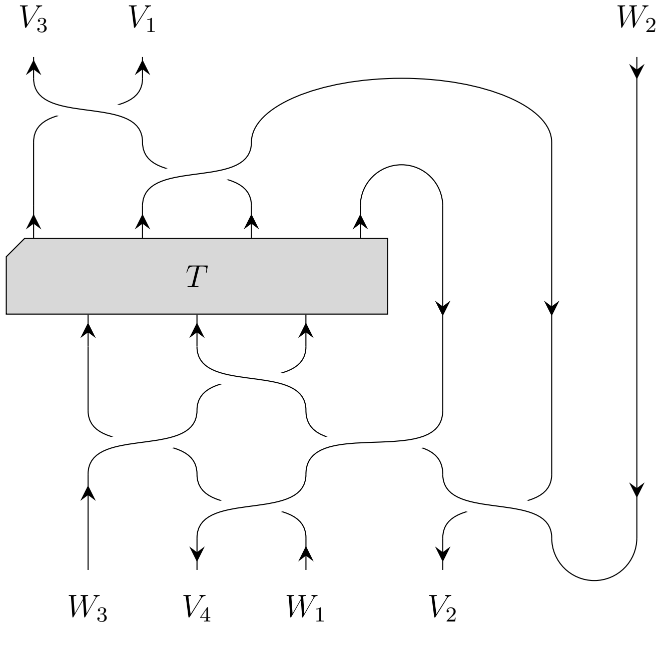
Bending lines involves communication between blocks; the new blocks $\tilde{T}_d$ have contributions from (potentially) many blocks T_c of the old tensor, namely from all those blocks whose coupled sector c is valid as the topmost inner sector of a fusion tree $a_1 \otimes \cdots \otimes a_{J+1} \to d$, i.e. such that $N_d^{c,a_{J+1}} > 0$ and $N_c^{a_1\dots a_J} > 0$. Unlike for the braiding which preserves entire rows, no units larger than tree blocks are preserved by this rearrangement. Therefore, the implementation for the general case that we propose in the following section may be used to realize the above results in practice.

5.3.8 Rearranging legs

We propose to support a broad but not fully general class of leg rearrangement that is specified by the following data. We specify a permutation of the legs, indicating for every leg in the domain and codomain of the original tensor, if it ends up in the domain or codomain of the result, and at which position. Additionally, we assign a unique height value to every leg, which specifies the chirality of the braid at every crossing; the leg with the higher height values goes over the other one. By coherence, every composite of braids, cups and caps that fulfills these constraints is equal, and we may choose any particular realization that is practical, in the implementation.

As an example, consider rearranging the legs of $T : W_1 \otimes W_2 \otimes W_3 \to V_1 \otimes V_2 \otimes V_3 \otimes V_4$ as follows.

$\tilde{T}$:=



(5.137)

We specify the target order of legs as a permutation π, which is split in two components π_i for the codomain ($i = 1$) and domain ($i = 2$) respectively. The above example is described by the permutation

$$\pi_1(V_1 \dots V_4, W_1^\star \dots W_3^\star) = (V_3, V_1, W_2^\star) \tag{5.138}$$

$$\pi_2(V_1^\star \dots V_4^\star, W_1 \dots W_3) = (W_3, V_4^\star, W_1, V_2^\star) \tag{5.139}$$

and height levels $(1, 4, 3, 7; 5, 6, 2)$, given in the leg order (figure 5.1), such that e.g. the W_3 leg has height value five. Thus, whenever they cross, it braids below W_2 which has the higher height value of six.

We *conceptually* decompose this rearrangement as a sequence of single braids and line bends, which are discussed in sections 5.3.6 and 5.3.7 respectively. We arbitrarily choose to start in the codomain, the other way around would be just as good.

1. Determine a permutation for the codomain, such that those legs that need to be bent to the domain are on the very right and those that should stay in the codomain are sorted according to their target positions. Apply a sequence of braids that achieves this permutation and has braid chiralities that are consistent with the height values.

2. Bend those legs from the codomain to the domain

3. Similar to step 1, apply braids to the domain such that those legs that should remain there appear in the target order and those that should be bent to the codomain are on the very right.

4. Bend those legs to the codomain

5. Perform a final permutation on the codomain, again similar to step 1, to bring those legs to their target positions.

We propose to programmatically derive an expression for the blocks of the resulting tensor, following the steps listed above and then acting on the blocks only once, instead of forming the blocks of the intermediate tensors after each step. The following framework allows us to treat both braids and bends on an equal footing and to compose them programmatically. Let us write multi-indices, such as e.g. $\varphi = (a_1 \dots a_J, \alpha, c, \beta, b_1 \dots b_K)$ that fully specify a pair of fusion and splitting tree α, β with matching coupled sector c, including their uncoupled sectors a_j and b_k. This multi-index uniquely identifies a tree block $\left[T_c\right]^{a_1 \dots a_J, \alpha}_{b_1 \dots b_K, \beta} =: T^{(\varphi)}$ of a tensor T. Now for both braiding and bending, the new tree blocks are linear combinations of the old tree blocks, with their m_j and n_k indices permuted according to the overall leg permutation π, that is

$$\left[\tilde{T}^{(\varphi)}\right]^{m_1 \dots m_J}_{n_1 \dots n_K} = \sum_\psi E^\varphi_\psi \cdot \left[T^{(\psi)}\right]^{\pi_1(m_1 \dots m_J)}_{\pi_2(n_1 \dots n_K)} \tag{5.140}$$

for some coefficients E^φ_ψ.

Similar to the braiding, expressions of this form can easily be composed, by composing the permutations π and matrix multiplication of the coefficients E, e.g. for two operations we have

$$\left[\tilde{\hat{T}}^{(\varphi)}\right]^{m_1\dots m_J}_{n_1\dots n_K} = \sum_{\zeta,\psi} \tilde{E}^{\varphi}_{\zeta}\cdot \hat{E}^{\zeta}_{\psi}\cdot \left[T^{(\psi)}\right]^{(\tilde{\pi}\circ\hat{\pi})_1(m_1\dots m_J)}_{(\tilde{\pi}\circ\hat{\pi})_2(n_1\dots n_K)}\ , \tag{5.141}$$

where the operation to be applied first has coefficients $\hat{E}$ and permutation $\hat{\pi}$. In practice, to exploit the sparsity, we propose the strategy transcribed in pseudocode in algorithm 5.1.

Algorithm 5.1 Building coefficients for leg rearrangement

Given a sequence of leg rearrangements with coefficients $(E_1), (E_2), \dots, (E_N)$, and a tree block index φ for the resulting tensor, compute the nonzero coefficients of the composite rearrangement. That is a mapping $\{\psi : E^{\varphi}_{\psi}\}$ from the tree block indices ψ of the original tensor that have a non-zero contribution in (5.140) to the respective non-zero coefficients $E^{\varphi}_{\psi} = \sum_{\zeta\chi\dots\xi}(E_N)^{\varphi}_{\zeta}(E_{N-1})^{\zeta}_{\chi}\dots(E_1)^{\xi}_{\psi}$. The order of this matrix product is such that we may think of the rearrangement with coefficients E_1 to be applied "first".

```
Initialize the mapping as $\mathcal{C} \leftarrow \{\varphi : 1\}$.
For every rearrangement step $n = N, \dots, 1$ in reverse order:
    Initialize an empty mapping $\mathcal{D} \leftarrow \{\}$.
    For every key $\phi$ in $\mathcal{C}$ and value $c = \mathcal{C}[\phi]$:
        For every $\zeta$ such that $(E_n)^{\phi}_{\zeta} \neq 0$ (see recipies below):
            If $\zeta$ is in $\mathcal{D}$, increment $\mathcal{D}[\zeta] \leftarrow \mathcal{D}[\zeta] + c\cdot(E_n)^{\phi}_{\zeta}$, else set $\mathcal{D}[\zeta] \leftarrow c\cdot(E_n)^{\phi}_{\zeta}$.
    Set $\mathcal{C} \leftarrow \mathcal{D}$.
Return $\mathcal{C}$
```

Let us summarize the results from the previous sections on braids and line bends in this language of E coefficients.

For braiding, e.g. in the codomain, we get nonzero contributions to a target tree pair $\varphi = (a_1 \dots a_J, \gamma, c, \beta, b_1 \dots b_K)$ of the form

$$E^{\varphi}_{\psi} = D^{\gamma}_{\alpha}(a_1 \dots a_J, c) \quad \text{for} \quad \psi = \big(\pi(a_1 \dots a_J), \alpha, c, \beta, b_1 \dots b_K\big) \tag{5.142}$$

and vanishing $E^{\varphi}_{\psi} = 0$ for all other ψ. Note that the D coefficients, as discussed in section 5.3.6, have further sparsity and further constrain the trees α that give nonzero contributions.

For bending a single line from the domain to the codomain, we get nonzero contributions to a target tree pair

$$\varphi = \big(a_1 \dots a_J, (e_1 \dots e_{J-2}, \mu_1 \dots \mu_{J-1}), c, (f_1 \dots f_{K-2}, \nu_1 \dots \nu_{K-1}), b_1 \dots b_K\big)$$

of the following form;

$$E^{\varphi}_{\psi} = \left[B^{c,\bar{a}_J}_{e_{J-2}}\right]^{\kappa}_{\mu_{J-1}} \quad \text{, where} \tag{5.143}$$

$$\psi=(a_1\dots a_{J-1},(e_1\dots e_{J-3},\mu_1\dots\mu_{J-2}),e_{J-2},(f_1\dots f_{K-2}c,\nu_1\dots\nu_{K-1}\kappa),b_1\dots b_K a_J)$$

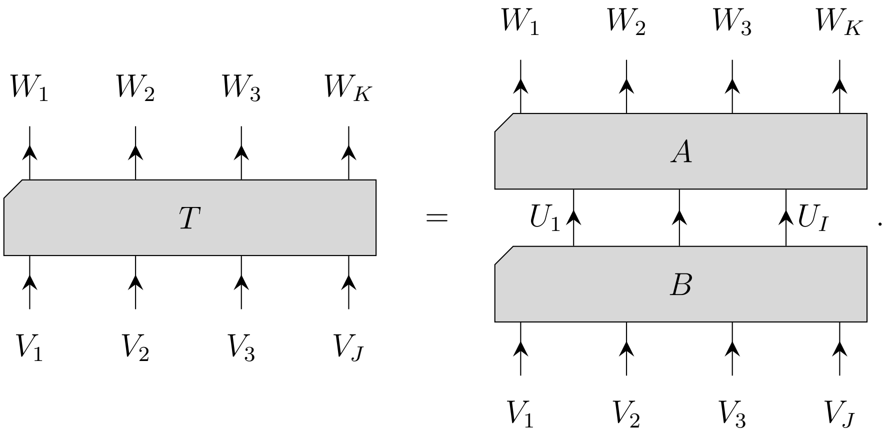
contains only $\kappa=1,\dots,N^{c,a_J}_{e_{J-2}}$ as a free parameter. The coefficients $E^{\varphi}_{\psi}=0$ vanish for all other ψ not of this form.

Let us reiterate at this point that leg rearrangements which involve only braids and no bends can be done by acting on entire rows / columns, which may be more efficient in practice.

5.3.9 Tensor contractions, inner product and norm

The most ubiquitous operation on tensors used in TNS methods is pairwise contraction of tensors. In our graphical notation, this means connecting any number of legs between the two tensors and leaving the rest open. This may require line bends (cups and caps) or braids to be introduced in the most general case.

Let us start with a simple case, the composition $T=A\circ B$ of two tensors $A:\mathcal{U}\to\mathcal{W}$ and $B:\mathcal{V}\to\mathcal{U}$. This is the special case of contractions where the tensors are already given with a convenient leg arrangement. That is graphically



(5.144)

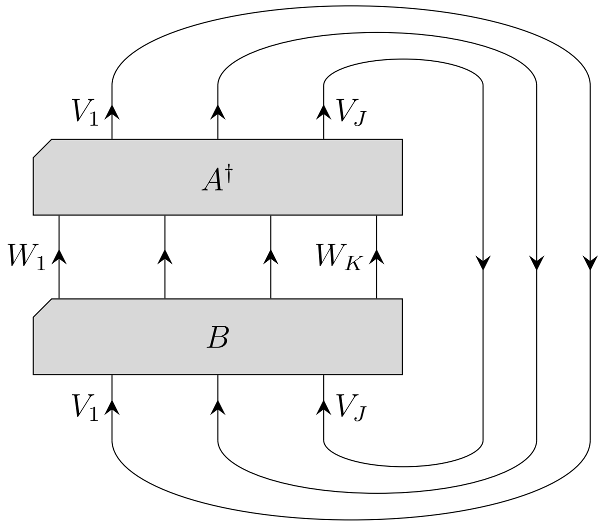
We find that the blocks of the result

$$\begin{aligned}[T_c]^{\kappa}_{\iota}\,\mathrm{id}_c &\overset{(5.110)}{=} \mathcal{X}^{\mathcal{W}}_{c,\kappa}\circ A\circ B\circ\mathcal{Y}^{\mathcal{V}}_{c,\iota} \overset{(5.109)}{=} \sum_{d,\lambda}\mathcal{X}^{\mathcal{W}}_{c,\kappa}\circ A\circ\mathcal{Y}^{\mathcal{U}}_{d,\lambda}\circ\mathcal{X}^{\mathcal{U}}_{d,\lambda}\circ B\circ\mathcal{Y}^{\mathcal{V}}_{c,\iota}\\ &\overset{(5.106)}{=} \sum_{\lambda}[A_c]^{\kappa}_{\lambda}[B_c]^{\lambda}_{\iota}\,\mathrm{id}_c=[A_c\cdot B_c]^{\kappa}_{\iota}\,\mathrm{id}_c\end{aligned} \tag{5.145}$$

are given by the blockwise matrix-matrix products.

Another special case for pairwise contraction is the “inner product”, where we contract all legs and the input tensors $A,B:\mathcal{V}\to\mathcal{W}$ are given with the same leg arrangement. Then, consider the Frobenius inner product

$$\mathrm{Tr}\left(A^\dagger \circ B\right) \quad = \tag{5.146}$$



.

The result is the scalar

$$\begin{aligned}\mathrm{Tr}\left(A^\dagger \circ B\right) &\overset{(5.109)}{=} \sum_{c,\lambda} \sum_{d,\kappa} \mathrm{Tr}\left(\mathcal{Y}^{\mathcal{V}}_{c,\lambda} \circ \mathcal{X}^{\mathcal{V}}_{c,\lambda} \circ A^\dagger \circ \mathcal{Y}^{\mathcal{W}}_{d,\kappa} \circ \mathcal{X}^{\mathcal{W}}_{d,\kappa} \circ B\right) \\ &\overset{(5.33),(5.106)}{=} \sum_{c,\lambda\kappa} [(A^\dagger)_c]^{\kappa}_{\lambda} [B_c]^{\lambda}_{\kappa} \ \mathrm{Tr}\left(\mathrm{id}_c\right) \overset{(5.119)}{=} \sum_c d_c \mathrm{Tr}\left((A_c)^\dagger \cdot B_c\right).\end{aligned} \tag{5.147}$$

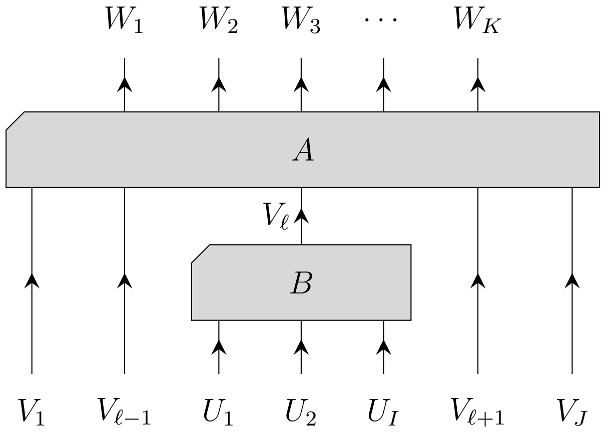
In words, the inner product of tensors is given by the weighted sum of blockwise matrix inner products, with the quantum dimensions of the coupled sectors as weights. As a corollary, we find that the Frobenius norm

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_c d_c \left\|A_c\right\|_{\mathrm{F}}{}^2} \tag{5.148}$$

is given by the weighted 2-norm of the blockwise Frobenius matrix norms.

Another special case that allows for an efficient implementation is when only one leg is contracted, which is the *only* leg in the codomain (domain) of the lower (upper) tensor. Considering the first of these cases where we contract the ℓ-th leg in the domain of $A : \bigotimes_j V_j \to \bigotimes_k W_k$ with the single leg in the codomain of $B : \bigotimes_i U_i \to V_\ell$, that is graphically

$$T \quad = \tag{5.149}$$



.

This case allows for a special implementation, since B has a trivial splitting tree in its decomposition (5.111), such that the fusion tree of B acts directly below the fusion tree of A and may be absorbed into a larger fusion tree via F moves.

First, for $I = 1$, that is if B only has a single leg in its domain, we find

$$\Big[[T_c]^{a_1\ldots a_J,\alpha}_{b_1\ldots b_K,\beta}\Big]^{m_1\ldots m_J}_{n_1\ldots n_K} = \sum_p \Big[[A_c]^{a_1\ldots a_J,\alpha}_{b_1\ldots b_K,\beta}\Big]^{m_1\ldots p\ldots m_J}_{n_1\ldots n_K} \Big[[B_{a_\ell}]^{a_\ell,1}_{a_\ell,1}\Big]^{m_\ell}_{p} \,. \tag{5.150}$$

Let us now assume $I \geq 2$. If $\ell = 1$, i.e. if we apply B to the very left domain leg of A, we do not even need to do F moves and directly find

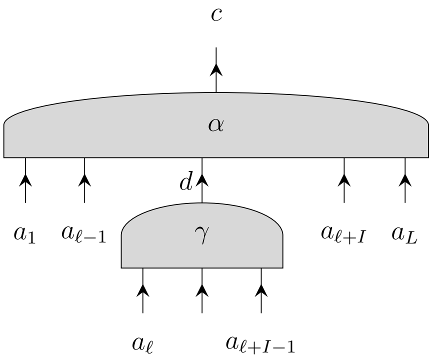
$$\Big[[T_c]^{a_1\ldots a_L,[\gamma-d-\alpha]}_{b_1\ldots b_K,\beta}\Big]^{m_1\ldots m_L}_{n_1\ldots n_K} = \sum_p \Big[[A_c]^{da_{I+1}\ldots a_L,\alpha}_{b_1\ldots b_K,\beta}\Big]^{pm_{I+1}\ldots m_L}_{n_1\ldots n_K} \Big[[B_d]^{a_1\ldots a_I,\gamma}_{d,1}\Big]^{m_1\ldots m_I}_{p} \,, \tag{5.151}$$

where $L = J + I - 1$ is the new number of legs in the domain and $[\gamma - d - \alpha]$ denotes a fusion tree index where d is inserted as an inner sector, that is with inner sectors $(f_1 \ldots f_{I-2}, d, e_1 \ldots e_{J-2})$ and multiplicity labels $(\nu_1 \ldots \nu_{I-1}, \mu_1 \ldots \mu_{J-1})$ where $\gamma = (f_1 \ldots f_{I-2}, \nu_1 \ldots \nu_{I-1})$ and $\alpha = (e_1 \ldots e_{J-2}, \mu_1 \ldots \mu_{J-1})$.

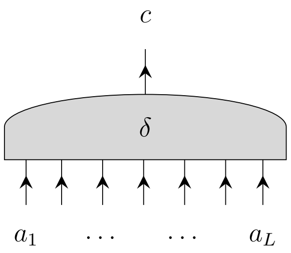
Finally, in the general case $\ell > 1$ we find

$$\begin{aligned}\Big[[T_c]^{a_1\ldots a_L,\delta}_{b_1\ldots b_K,\beta}\Big]^{m_1\ldots m_L}_{n_1\ldots n_K} &= \sum_{\gamma d\alpha p} F^{\gamma d\alpha}_{\delta}(a_1 \ldots a_L, c)\times \\ \times \Big[[A_c]^{a_1\ldots a_{\ell-1},d,a_{\ell+I}\ldots a_L,\alpha}_{b_1\ldots b_K,\beta}\Big]^{m_1\ldots m_{\ell-1},p,m_{\ell+I}\ldots m_L}_{n_1\ldots n_K} &\Big[[B_d]^{a_\ell\ldots a_{\ell+I-1},\gamma}_{d,1}\Big]^{m_\ell\ldots m_{\ell+I-1}}_{p} \,,\end{aligned} \tag{5.152}$$

where the coefficients are generalized F symbols which arise as



$$= \sum_{\delta} F^{\gamma d\alpha}_{\delta}(a_1 \ldots a_L, c)$$



(5.153)

and can be written as a contraction of a sequence of $I - 1$ single F symbols (5.70).

More general tensor contractions can be performed by first permuting legs, according to section 5.3.8, and subsequently contracting as one of the special cases listed here.

5.3.10 Tensor decompositions

For a tensor $T : \mathcal{V} \to \mathcal{W}$, we can obtain an SVD as follows. Define a new space

$$V_S := \bigoplus_{c \in \mathcal{S}} \bigoplus_{m=1}^{\min(N_c^{\mathcal{V}}, N_c^{\mathcal{W}})} c \tag{5.154}$$

and define tensors $U : V_S \to \mathcal{W}$, $S : V_S \to V_S$ and $V^\dagger : \mathcal{V} \to V_S$ via blockwise SVD, that is such that $T_c =: U_c \cdot S_c \cdot (V_c)^\dagger$. The defining properties of an SVD follow directly from (5.145), (5.119) and (5.118). Namely, we have $T = U \circ S \circ V^\dagger$, as well as $U^\dagger \circ U = \mathrm{id}_{V_S} = V^\dagger \circ V$ and S is diagonal with real positive entries.

Similarly, any other decomposition of symmetric tensors whose defining properties – such as here composition, identities and the dagger – go through to the block level, can be carried out simply by applying the respective matrix decomposition to each block. In particular, this includes the (hermitian) eigendecomposition and the polar decomposition. The QR and LQ decompositions work except for the caveat that the upper (lower) triangular property of the R (L) factors only holds on the block level, it is not a meaningful property of the resulting symmetric tensor.

Note that there are additional considerations for optimal truncation of an SVD, that are not relevant in the abelian case of algorithm 2.8. Since the square norm of the tensors is a *weighted* sum of blockwise square norms – see (5.148) – we need to consider the weights when truncating. In particular, discarding a singular value $\lambda = (S_c)_{ii}$ and corresponding columns of the isometries U_c, V_c in the matrix SVD of a block T_c, results in a truncation error $\|T - \tilde{U} \circ \tilde{S} \circ \tilde{V}^\dagger\|_\mathrm{F} = \sqrt{d_c}\lambda$. Thus, when selecting which singular values in the entire S tensor to discard, they should be prioritized such that those with the smallest value of $\sqrt{d_c}(S_c)_{ii}$ are discarded first. For an intuition, consider a SU(2) symmetry, where discarding a single singular value from a block with coupled sector given by the spin S irrep is equivalent to discarding a multiplet of $d_S = 2S + 1$ degenerate singular values in a non-symmetry conserving representation of the tensor.

5.4 Remarks on implementation

An implementation of the framework for symmetric tensors discussed in this chapter is under active development, publicly on GitHub[10], where the current prototype is open-source and publicly available. This is part of a rework of the tensor backend handling linear algebra routines of symmetric tensors, planned for the next major release of the TeNPy library. In this section, we list a few additional considerations regarding this implementation.

[10] See the repository https://github.com/tenpy/tenpy, and in particular the pull request https://github.com/tenpy/tenpy/pull/309, which will inevitably become outdated at some point, but should remain a good starting point to track down up to date links. Alternatively, see https://github.com/Jakob-Unfried/phd_thesis.

5.4.1 Diagonal Tensors

We propose to use a dedicated implementation for a special class of tensors, the diagonal tensors. A *diagonal tensor* is a tensor $D : V \to V$ from a single space (no tensor product) to itself, such that its blocks are diagonal

$$[[D_c]^{c,1}_{c,1}]^m_n =: D_{c,m}\delta_{m,n}. \tag{5.155}$$

Here, we have already used in the notation that the uncoupled sectors must equal the coupled sectors, and there is only a single trivial fusion tree for that mapping, which is indexed by 1. The singular values from an SVD, as well as the eigenvalues from a (hermitian) eigendecomposition, have this property and are the main motivation for introducing diagonal tensors.

As a result, we may store only the diagonal entries $D_{c,m}$ and allow cheaper implementations for contraction with other tensors. Additionally, we may unambiguously and straightforwardly implement elementwise operations on these diagonal tensors, such as taking powers, the square root, and so on.

5.4.2 Charged Tensors

It is convenient to introduce the concept of a charged tensor. This generalizes the idea of charged tensors as formulated for the abelian group case in (2.69) to the general categorical case. We proposed to think about the symmetric tensors as symmetric maps $T : I \to \mathcal{V} := \bigotimes_n V_n$ from the trivial sector to the tensor product of the *legs* $V_1, \dots, V_N$. On the other hand, a charged tensor (with the same legs) is a symmetric map $T : C \to \mathcal{V}$ where the symmetry space C describes its charge. This contains the notion (2.69) of a charged tensor as a special case, namely if the symmetry is an abelian group and C a one-dimensional irrep. For the general notion of a charged tensor, however, we do not impose any requirements on C.

Thus, a charged tensor $\tilde{T}$ with legs given by the factors of $\mathcal{V}$ is described by a symmetric tensor $T : C \to \mathcal{V}$ that has one additional leg $C^\star$. For group symmetries, where C is a vector space, we may additionally specify a state $|\gamma\rangle \in C$ on that leg and view the composite object given by T and $|\gamma\rangle$ as the charged tensor. Such a composite is not itself a symmetric map, but we can use the framework for symmetric tensors to manipulate it by acting on its "symmetric part" T.

Let us consider as a concrete example a system with a U(1) group symmetry conserving the S^z magnetization of a spin-$\frac{1}{2}$ chain. The sectors $\mathcal{S} = \{V_\mathbf{a} | \mathbf{a} \in \mathbb{Z}\}$ are labeled by integers which represent the S^z magnetization of that sector in units of $\hbar/2$. The two-dimensional local Hilbert space decomposes into sectors as $H = V_\mathbf{1} \oplus V_\mathbf{-1}$ in the z basis. Now consider the raising and lowering operators $\sigma^+ = |\uparrow\rangle\langle\downarrow|$ and $\sigma^- = |\downarrow\rangle\langle\uparrow|$. As tensors in $\mathcal{V} = H \otimes H^\star$, or as maps $H \to H$, they are not symmetric. As a consequence – or as the most intuitive way of seeing this – they do not commute with the conserved charge S^z. They can, however, be written as charged tensors, e.g. $\Sigma^+ : V_{+\mathbf{2}} \to H \otimes H^\star, |\phi\rangle \mapsto \langle\Uparrow|\phi\rangle\sigma^+$, where we have written $|\Uparrow\rangle$ for the only state in an orthonormal basis of the charge-leg C. Now by choosing

$|\gamma\rangle = |\Uparrow\rangle$, we can recover the original operator as $\Sigma^+(|\Uparrow\rangle) = \sigma^+ \in H \otimes H^\star$. The lowering operator can be similarly written as a charged tensor with $C = V_{\mathbf{-2}}$.

For a slightly less trivial example, let us consider the same physical system and the operator $\sigma^x = (\sigma^+ + \sigma^-)/2$. As a tensor in $H \otimes H^\star$ it is clearly not symmetric either. It can, however, be written as a charged tensor with symmetric part

$$X : C \to H \otimes H^\star, |\phi\rangle \mapsto \langle\Uparrow|\phi\rangle\, \sigma^+ + \langle\Downarrow|\phi\rangle\, \sigma^-, \tag{5.156}$$

where the charge leg is $C = V_{\mathbf{+2}} \oplus V_{\mathbf{-2}}$, and we can recover $\sigma^x = X(|\gamma\rangle)$ with the charged state $|\gamma\rangle = (|\Uparrow\rangle + |\Downarrow\rangle)/2$. Note that the charged state is not symmetric under the U(1) symmetry, and in fact, C does not contain any symmetric states other than 0.

This perspective on charged tensors allows us to use them also for quantum states with fermionic or anyonic grading. See appendix A for the example categories we use here. With the category **Ferm**, for example, a charged tensor whose charge leg is the sector of odd fermionic parity can be used to describe a quantum state with an odd number of fermions. Conversely, with the category **Fib** of Fibonacci anyons, e.g., considering a golden chain [173] system, a charged tensor with the τ sector as its charged leg can describe a state of multiple sites that is in the τ topological sector. These are precisely the two-body states that are energetically favored by a single local term in the golden chain Hamiltonian.

An alternative perspective is that we may use the concept of a charged tensor to effectively hide one leg of a tensor from outside algorithms. Note, for example, that if we take the symmetric part X of the σ^x operator from the above U(1) symmetric example and compose it with its dagger, we obtain a tensor $X^\dagger \circ X$ that is equivalent (by re-ordering the legs) to the two-body operator $(\sigma^x \otimes \sigma^x)/4$. We can use this, e.g., to evaluate a correlation function $\langle\psi|\sigma^x_i(t)\sigma^x_j|\psi\rangle$ in a symmetric MPS $|\psi\rangle$, where time dependence is w.r.t. the dynamics induced by a symmetric Hamiltonian, even though the single operator σ^x_j is not symmetric. To do this, apply the symmetric part of the charged tensor X_j to the MPS $|\psi\rangle$, leaving an extra charge leg C on one of the tensors. We may then perform time evolution using one of the methods discussed in section 2.3 to find an MPS representation of $\mathrm{e}^{-\mathrm{i}Ht}\sigma^x_j|\psi\rangle$ and finally contract the dangling leg with its partner on σ^x to evaluate the correlation function at time t.

5.5 Benchmarks

Let us now study the benefits of using the symmetry backend in a benchmark of the prototype implementations developed for a future version of TeNPy. We provide the benchmark code, with a pinned version of the prototype implementation, publicly on GitHub[11].

We compare the following three tensor backends, that implement the storage format of tensors and operations on them. First, the *trivial backend* simply stores all

[11] https://github.com/Jakob-Unfried/phd_thesis

$\prod_n \dim V_n$ entries of a tensor $T \in \bigotimes_n V_n$ as a dense array and can not exploit/enforce any symmetries. In the implementation, this is essentially a thin wrapper around numpy. Secondly, an *abelian backend* that can exploit abelian symmetry groups, as discussed in section 2.5. This is very similar to the implementation in tenpy version 1 [4], though the prototype is less performant, since it is implemented in pure python, without any compilation. Lastly, a *fusion tree backend* that uses the storage format and manipulations as introduced in this chapter.

At the time of writing, the concrete implementations in the prototype are called `NoSymmetryBackend`, `AbelianBackend` and `FusionTreeBackend`, respectively. Note that the implementations are prototypes that have not yet been optimized for performance.

5.5.1 Tensor contraction

Let us first benchmark tensor contraction. We generate SU(2) symmetric test tensors with pre-determined legs and random free parameters as follows. We generate a leg V as the N-fold tensor product of a spin-$\frac{1}{2}$ Hilbert space, that is $V = \bigotimes_{n=1}^{N} \mathcal{H}_{1/2}$. This defines a somewhat realistic distribution of SU(2) sectors, as this is the virtual leg that an MPS needs to have to represent any state on a chain of $2N$ spin-$\frac{1}{2}$ sites exactly, meaning without truncation. In particular, the largest sector is spin $\frac{N}{2}$. We generate four-leg symmetric tensors $A, B : V \otimes V \to V \otimes V$ by populating their free parameters (5.106) with reproducible (pseudo-)random numbers, from a numpy random generator with a fixed seed. Then, we perform a timing benchmark of the contraction $A \circ B$, contracting two pairs of legs. For each of the three backends, we either enforce the full SU(2) symmetry (if supported), the U(1) subgroup that only conserves S^z (if supported), or do not enforce any symmetry. We run the benchmark on a single core of an Intel Core i7-6700 CPU. The results are shown in figure 5.3.

We find that if no symmetry is enforced, all backends show nearly identical performance, as the implementation eventually delegate to a single matrix-matrix multiplication of the respective single block of each tensor. We find the expected scaling $\sim (\dim V)^6$ of multiplying $(\dim V)^2 \times (\dim V)^2$ matrices, as soon as the scaling regime is reached at $\dim V \gtrsim 16$. At smaller sizes, overhead from the pure-python bookkeeping of tensors and their legs contributes significantly.

For the abelian backend, we chose the “standard” form of a four-leg tensor that stores separate blocks for every combination of four individual charges, one per leg. Thus, when contracting multiple legs, the combinatorics of which pairings of smaller blocks contribute, results in a sizeable overhead, even when the combinatorics is trivial, in the case where nothing is conserved. We expect this overhead to be reduced significantly in an optimized implementation using a compiled language. In practice, if the same tensor is used in multiple contraction calls, as e.g. in a Lanczos eigensolver, the combinatorics can be done once, ahead of time, by combining the legs.

We expect the same scaling behavior $\mathrm{O}((\dim V)^6)$ when enforcing the U(1) symmetry. Consider the following heuristic intuition. Assume, for simplicity, that after

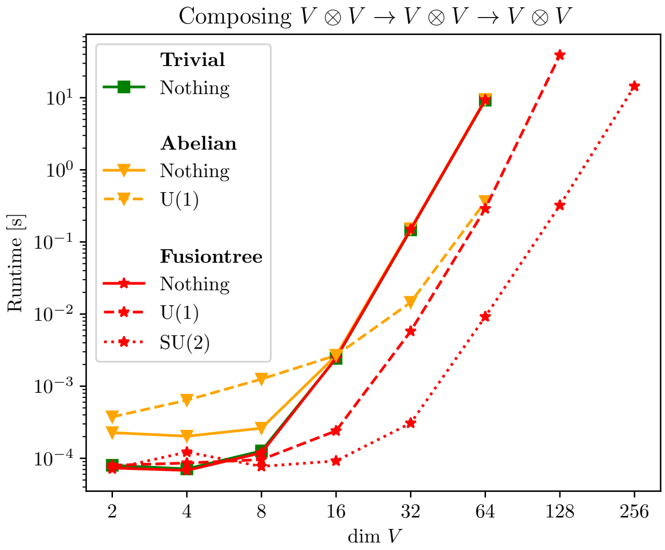


Figure 5.3: Runtime for contracting two common legs between two four-leg tensors averaged over several RNG seeds. For each sample size (horizontal) axis and RNG seed, we generate the same SU(2) symmetric data and then benchmark the decomposition for the following different cases. We choose to either enforce the full SU(2) symmetry (dotted lines), the abelian U(1) subgroup (dashed lines), or no symmetry (solid lines). We compare the trivial backend (green squares), an abelian backend (yellow triangles) as discussed in section 2.5, and a fusion tree backend (red stars), as discussed in this chapter, for the symmetry cases that they support.

reshaping the tensors to $n \times n$ matrices with $n = (\dim V)^2$, they consist of k blocks of equal size n/k. Then, the blockwise matrix product can be carried out at a cost of $kT(k/n) \sim n^3/k^2$, where $T(n) \sim n^3$ denotes the cost of forming matrix-matrix products of $n \times n$ matrices.

For the fusion tree backend, no additional combinatorics arises from the fact that we contract multiple legs, and this particular contraction is implemented directly as blockwise matrix-matrix products. This is less flexible, however, since the abelian backend implementation can perform any other contraction of any two leg pairs in the same way, with comparable overhead, while the fusion tree backend would need to perform leg manipulations, such as braids or line bends first. These would introduce a noticeable, but subdominant additional cost. Therefore, the chosen scenario where the tensors are already given with a leg arrangement that allows the contraction to be carried out directly as composition is favorable for the fusion tree backend. It highlights only the algorithmic step where enforcing the full SU(2) symmetry gives the most net benefit.

We find a substantial speedup when enforcing the full SU(2) symmetry, which motivates the development and use of the tensor backend.

5.5.2 Singular value decomposition

Next, we perform a similar benchmark of the singular value decomposition (SVD). We generate random test tensors $A : V \to V$ that have the full SU(2) symmetry, whether we enforce it explicitly or not, with two equal legs V, each with sectors as described in the previous section. We perform a timing benchmark of computing the SVD of A, with the same combinations of backend and enforced symmetry as above. The results are shown in figure 5.4.

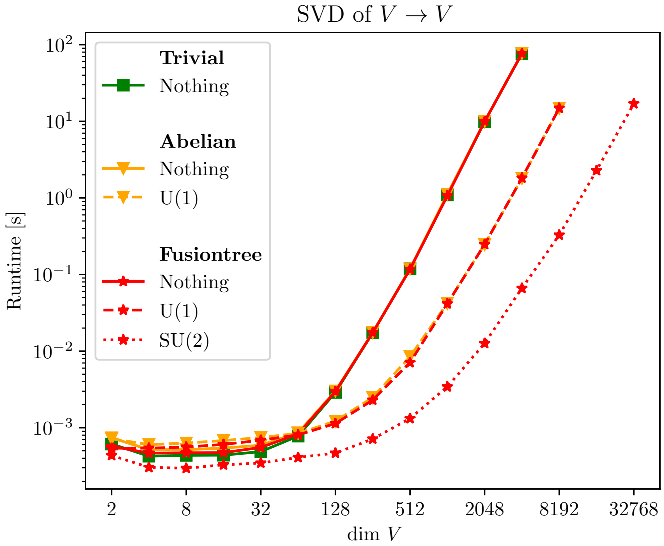


Figure 5.4: Runtime of computing an SVD of a two leg tensor, averaged over several RNG seeds. The setup is the same as for the contraction benchmark of figure 5.3, except we generate only a single two leg tensor $V \to V$.

Similar to contraction, we find that overhead is dominant for small samples $\dim V \lesssim 128$, crossing over to a scaling regime where we clearly see substantial speedups for exploiting the U(1) subgroup and again for exploiting the full SU(2) symmetry. We find the agreement with the expected scaling $\sim (\dim V)^3$ in all cases and see that if the same symmetry is enforced, the performance of different backends matches as soon as the scaling regime is reached.

5.6 Conclusion

In this chapter, we have provided a mathematical framework for symmetric tensors that allows us to enforce abelian and non-abelian symmetries and covers tensors that live in a more general tensor category, e.g. with the statistics of fermionic or anyonic degrees of freedom. We have defined a graphical language for the basic concepts, identified the free parameters of symmetric tensors, and developed in detail how to perform operations on the tensors, such as contractions, leg manipulations, and factorizations. This may serve as an implementation guide for a tensor backend and is the basis of the prototype implementation, which is publicly available in the TeNPy repository [4].

For the group case, we have demonstrated the speedups that can be obtained by exploiting non-abelian groups in basic tensor algebra operations, compared to only enforcing the largest abelian subgroup, which is possible with an abelian tensor backend. The general categorical case allows simulations of many-body quantum states of degrees of freedom with non-trivial exchange statistics, such as fermions or anyons, by enforcing the respective grading on the tensor level.

Future directions for the concrete implementation in TeNPy are performance optimization and integration into the rest of the library. As a first step towards interoperability in the vast landscape of tensor algebra libraries, we propose to establish a unified storage format of symmetric tensors. In the context of tensor network methods, this may eventually be extended to unified storage formats for the common classes of tensor network states, such as e.g. MPS. An additional avenue for future development in this framework is how to incorporate generalized, non-invertible symmetries [174].

Chapter 6

Conclusion

In this thesis, we focused on tensor network methods for simulating quantum many-body systems and discussed several algorithmic advancements. We started with a pedagogical review of tensor networks, focusing on MPS methods such as the TEBD, DMRG, and MPO evolution algorithms, as well as PEPS for two-dimensional systems. We reviewed how to enforce abelian symmetries on the tensor level, exploit the resulting block sparse structure, and introduced the TeNPy library, implementing symmetric linear algebra and the MPS algorithms.

We then discussed alternative approximate low-rank factorizations that can replace the SVD as a truncation step in tensor network simulations, highlighting as an example use case the application in the TEBD algorithm. We proposed a QR-based truncation method and discussed its conceptual relation to randomized linear algebra. We demonstrated an improved scaling with the dimension of the local Hilbert space from cubic to quadratic in a benchmark. We found that as an SVD-free algorithm, we can obtain significant speedups on GPU hardware that are not possible for the SVD-based version. Future directions include incorporating these faster and GPU-friendly truncation steps into broader algorithmic settings, as well as developing (randomized versions of) the QRCP or QLP decomposition for symmetric tensors.

Next, we proposed a gradient-based approach for optimizing finite PEPS for ground state search or time evolution. We were, at this point, unable to produce an algorithm that brings the success of gradient-based optimization in iPEPS to finite systems. We were, however, able to shed some light on how well other PEPS ground state searches exhaust the variational power of the ansatz class – by finding better ground state approximations at the same bond dimension. Moreover, to our knowledge, the resulting time evolution algorithm is the only method to simulate dynamics using *finite* PEPS that produces results of useable accuracy. As such, technical improvements of the method, and in particular its stability and performance should be pursued.

Finally, we compiled a mathematical framework that allows the enforcement of non-abelian symmetries on the tensor level in terms of parametrization of symmetric tensors. The same framework – that of monoidal category theory – allows us to also represent tensors with the statistics of fermions or anyons and can be used to build tensor network states for systems with fermionic or anyonic degrees of freedom. We proposed a storage format of symmetric tensors and derived in detail how to implement common linear algebra operations in terms of the stored free parameters. A functioning prototype – not yet optimized for performance – was developed by the author and is publicly available. With a simple benchmark of the prototype we demonstrated the speedups that motivate exploiting the full non-abelian symmetries in models that have them. An implementation of this machinery is under active development, with the aim of being incorporated in the TeNPy library at the next major release. It will enable speedups from enforcing larger non-abelian symmetry groups, charge pumping experiments for breaking of a non-abelian symmetry, and simulation of anyonic systems.

A common theme throughout the independent topics in the separate chapters seems to be that fixing gauge freedoms may be prohibitive and should not be done without reason, and it may be worthwhile to consider relaxing to weaker requirements. In the context of MPS, this means relaxing from the full canonical form to only an isometric form with non-diagonal bond matrices, while in the context of approximate low-rank factorizations, this means allowing deformed factorizations, where the central matrix is not diagonal. This relaxation to a weaker, e.g. isometric form, is not new algorithmically, as subspace expansion methods in DMRG typically result in non-diagonal bond matrices. In either case, when we want to do truncation, we (mostly) only care about identifying a particular subspace for truncation that admits a good approximation of the tensor network state or of the matrix we want to factorize. While this subspace is crucial, a particular choice of basis for this space – in which, e.g., the MPS bond matrix becomes diagonal – is secondary. Moreover, enforcing the very particular basis, e.g. of singular vectors in case of a (truncated) SVD, may introduce divergent terms in automatic differentiation (AD) which are not present for the weaker deformed singular value decomposition (dSVD). Additionally, relaxing such requirements admits the alternative factorization methods, such as the QR-based truncation scheme, ultimately enabling hardware acceleration to be used.

Incremental technical advances of tensor network methods, as described in this thesis and proposed for future development both in this conclusion and in more detail in the per-chapter conclusions, push the capabilities of simulations. These simulations allow direct access to the properties of model systems and, as such, are invaluable tools in the quest for a better theory of superconductivity in the cuprates, in the study of spin liquids, of topological order, and many more exotic phenomena in condensed matter systems and beyond.

Appendix A

Topological data for common symmetries

In this chapter we provide extra material for the discussions regarding symmetries, in section 2.5 and chapter 5. In section A.1, we review the basics of group representations, as well as prominent results such as Schur's lemma. In the subsequent sections, we discuss common symmetry groups, and give the topological data needed to enforce these symmetries in the framework of chapter 5, using fusion trees and their manipulations. We cover the abelian groups $\mathbb{Z}_N$ and U(1) in section A.2 and A.3, the nonabelian group SU(2) in section A.4, fermionic grading in A.5 and Fibonnaci anyons in section A.6. We discuss combining symmetries in section A.7.

A.1 Review: Representation theory

Let us review some basics from the representation theory of groups.

A (faithful) *representation* U of a group G on a vector space V is a group homomorphism $U : G \to \mathrm{Hom}\,(V,\ V)$. This means a group representation assigns to every group element $g \in G$ a linear map $U(g) : V \to V$, such that the group structure is preserved, i.e. for $g, h \in G$ we have $U(gh) = U(g)U(h)$. We suggestively use the symbol U for the representation, since we want to think of it as unitary.

A linear map $f : V \to W$ is *equivariant* between representations U on V and U' on W if $f \circ U(g) = U'(g) \circ f$ for all $g \in G$. Two representations U and $\tilde{U}$ of the same group G on vector spaces V and $\tilde{V}$ respectively, are *equivalent* if there is an equivariant vector space isomorphism $S : V \to \tilde{V}$, i.e. an invertible linear map such that $\tilde{U}(g) = S \circ U(g) \circ S^{-1}$ for all $g \in G$. In that case, we write $U \cong \tilde{U}$. We can think of equivalence as the criterion for a basis change between vector spaces $V \to \tilde{V}$ with associated representations, such that equivariance is preserved, meaning if $f : V \to W$ is equivariant, so is $f \circ S^{-1}$.

A representation U on a vectorspace V is *reducible*, if there is a non-trivial proper subspace $0 \neq W \subset V$ which is invariant under U, that is if $U(g)(w) \in W$ for all

$w \in W$ and $g \in G$. The trivial subspace $W = 0$ and the full space itself $W = V$ are always invariant and are therefore excluded. A representation is *irreducible* if there is no such W. We refer to irreducible representations as *irreps* for short. While reducibility is suggestively named, we are not in general guaranteed that they can actually be reduced. Let us first characterize what it means to be "reduced", namely as a *direct sum*.

The direct sum $V \oplus W$ of vector spaces V, W is given by $\{(v, w) | v \in V, w \in W\}$ with componentwise addition and scalar multiplication. If V, W are inner product spaces, we can equip $V \oplus W$ with the inner product $\langle (v_1, w_1) | (v_2, w_2) \rangle = \langle v_1 | v_2 \rangle_V + \langle w_1 | w_2 \rangle_W$. The direct sum $f \oplus g$ of linear maps $f_1 : V_1 \to V_2$ and $g : W_1 \to W_2$ is given by componentwise application $f \oplus g : V_1 \oplus W_1 \to W_1 \oplus W_2, (v, w) \mapsto (f(v), g(w))$ and in an appropriate basis, its matrix representation is the block-diagonal matrix formed from the matrix representations of f, g. Finally, the direct sum $U \oplus \tilde{U}$ of representations U and $\tilde{U}$ of a group G on vector spaces V and $\tilde{V}$ respectively is a representation on $V \oplus \tilde{V}$ given by $(U \oplus \tilde{U})(g) = U(g) \oplus \tilde{U}(g)$, i.e. by the componentwise direct sum of linear maps. Generalizations of these definitions to direct sums of multiple spaces / maps / representations are straight-forward.

Irrep decomposition: We can now reduce general representations, if we include the additional assumption that they are unitary; Any unitary representation U of a group G on a finite-dimensional vector space V is equivalent to a finite direct sum $U \cong \bigoplus_{n=1}^{N} U_n$ of irreps U_n of G.

Proof: If U is irreducible, the claim is trivially fulfilled. Let us now assume U is reducible and let $0 \neq W \subset V$ be an invariant subspace.

Consider the orthogonal complement $W^\perp = \{v \in V | \langle v | w \rangle = 0 \ \forall w \in W\}$. For any $g \in G$, $U(g)$ is unitary and in particular injective. Thus the restriction $U_1(g) : W \to W, w \mapsto U(g)(w)$ is (a) well-defined since W is invariant and (b) also injective. Since it is a linear map between vector spaces of equal finite dimension, it is also surjective. Now let $x \in W^\perp$ and $w \in W$. Because of surjectivity, there is a $w' \in W$ s.t. $U(g)(w') = w$, which means that $U(g)(x)$ is in $W^\perp$ since $\langle U(g)(x) | w \rangle = \langle U(g)(x) | U(g)(w') \rangle = \langle x | w' \rangle = 0$. This shows invariance of $W^\perp$ and allows us to define the restriction $U_2(g) : W^\perp \to W^\perp, x \mapsto U(g)(x)$. Both restrictions $U_{1/2}$ are unitary by construction.

Now consider the vector space isomorphism $S : V \to W \oplus W^\perp, v \mapsto (P_W(v), v - P_W(v))$, where P_W is the projector on the subspace W. The inverse is simply $S^{-1} : W \oplus W^\perp \to V, (w, w') \mapsto w + w'$. We can conclude equivariance for all $g \in G$

$$\begin{aligned}(S \circ U(g) \circ S^{-1})(w, w') &= S \circ (U(g)(w) + U(g)(w')) = S \circ (U_1(g)(w) + U_2(g)(w')) \\ &= (U_1(g)(w), U_2(g)(w') = (U_1 \oplus U_2)(g)(w, w') \ ,\end{aligned} \tag{A.1}$$

where we used that $U(g)(w) = U_1(g)(w)$ for $w \in W$ and similarly for $U(g)(w) = U_2(g)(w')$ for $w \in W^\perp$. This establishes $U \cong U_1 \oplus U_2$.

This procedure can now be iterated for U_1 and/or U_2, until all components are irreducible. This terminates after a finite number of steps, as the dimension of the representation spaces starts at the finite $\dim V$ and strictly decreases at each step. □

The irrep decomposition is in general not unique, as any irrep U_n may be replaced by an equivalent irrep $\tilde{U}_n \cong U_n$. It is therefore convenient to choose one representative for each equivalence class of irreps. We write $U_{\mathbf{a}}$ for such a representative and call it a *sector*. For any group, the *trivial representation* $U_{\mathbf{0}} : g \mapsto 1$ on the one-dimensional space $\mathbb{C}$ is irreducible, and can be chosen as the representative of the *trivial sector*, also called the symmetric sector. Let us write $\mathcal{S}_G$ for the set of sector labels such that every irrep U of a group G has exactly one $\mathbf{a} \in \mathcal{S}_G$ such that $U \cong U_{\mathbf{a}}$. Let us consider a few examples, which we derive and discuss in more detail in the following sections. We find integer labels $\mathcal{S}_{\mathrm{U}(1)} = \mathbb{Z}$ for U(1), and $\mathcal{S}_{\mathbb{Z}_N} = \mathbb{Z}_N$ for $\mathbb{Z}_N$. For multiple symmetries, i.e. for the product group $G \times H$ we find $\mathcal{S}_{G\times H} = \mathcal{S}_G \times \mathcal{S}_H = \{(\mathbf{a}, \mathbf{b}) | \mathbf{a} \in \mathcal{S}_G, \mathbf{b} \in \mathcal{S}_H\}$, i.e. tuples of the respective individual sector labels.

Now, unitarity is not a strong requirement. If G is a finite group, a compact Lie group, or any other group that admits a right-invariant Haar measure, any of its representations D on a space V is equivalent to a unitary representation U on the same representation space V.

Proof: Let D be a representation of G on V and $\langle - | - \rangle$ the inner product on V. First, we construct a different inner product $\langle - | - \rangle_D$ on V w.r.t. which D is unitary. Under the assumptions above, we have some integral measure $\int_G \mathrm{d}x$ on the group G, which is e.g. given by $\int_G \mathrm{d}x f(x) = \frac{1}{|G|} \sum_{x\in G} f(x)$ for finite groups G and group functions $f : G \to \mathbb{C}$. It is a (right-) invariant measure, meaning $\int_G \mathrm{d}x f(xy) = \int_G \mathrm{d}x f(x)$ for any $y \in G$. Using this measure we define a different inner product for $v, w \in V$ as

$$\langle v|w\rangle_D := \int \mathrm{d}x \, \langle D(x)(v)|D(x)(w)\rangle . \tag{A.2}$$

It is easy to check that it is indeed an inner product (i.e. hermitian, linear and positive). Now for any $g \in G$ and $v, w \in V$ we have

$$\langle D(g)(v)|D(g)(w)\rangle_D = \int \mathrm{d}x \, \langle D(xg)(v)|D(xg)(w)\rangle = \langle v|w\rangle_D \tag{A.3}$$

because of the invariance of the measure, meaning D is indeed unitary w.r.t. $\langle - | - \rangle_D$.

Now, there is an isomorphism $S : V \to V$ of inner product spaces $(V, \langle - | - \rangle_D)$ and $(V, \langle - | - \rangle)$, meaning $\langle S(v)|S(w)\rangle = \langle v|w\rangle_D$. It can e.g. be constructed by mapping an orthonormal basis in $(V, \langle - | - \rangle)$ to an orthonormal basis in $(V, \langle - | - \rangle_D)$ and linearly extending. Define the representation U of G on V via $U(g) := S \circ D(g) \circ S^{-1}$, which is equivalent to D by construction. Now for $g \in G$ and $v, w \in V$ we have

$$\begin{aligned}\langle U(g)(v)|U(g)(w)\rangle &= \left\langle (D(g) \circ S^{-1})(v) \middle| (D(g) \circ S^{-1})(w) \right\rangle_D = \left\langle S^{-1}(v) \middle| S^{-1}(w) \right\rangle_D \\ &= \langle v|w\rangle ,\end{aligned} \tag{A.4}$$

where we used the defining property of S, that D is unitary w.r.t. $\langle -|-\rangle_D$ and again the defining property of S. Thus we find that U is unitary w.r.t. the inner product $\langle -|-\rangle$ on V and equivalent to D. □

Thus for finite groups or compact Lie groups, which covers virtually all cases for symmetry groups in condensed matter physics, we may simply assume a representation is unitary, up to equivalence. As a corollary, *any* representation – unitary or not – of a finite group or a compact Lie group is equivalent to a direct sum of irreps.

The main result from representation theory that allows an efficient representation of symmetric tensors is *Schur's Lemma*, which comes in two parts.

Schur's lemma part 1: Let U and $\tilde{U}$ <u>in</u>equivalent irreps of a group G on finite dimensional vector spaces V and $\tilde{V}$ respectively. Now if $M : V \to \tilde{V}$ is equivariant, meaning linear and $\tilde{U}(g) \circ M = M \circ U(g)$ for all $g \in G$, then $M = 0$.

Proof: Distinguish three cases regarding dimensionality.

In case 1, assume $\dim V < \dim \tilde{V}$. Consider the subspace $M(V) := \{M(v)|v \in V\} \subseteq \tilde{V}$ and let $x = M(v) \in M(V)$. Then $\tilde{U}(g)(x) = (M \circ U(g))(v) \in M(V)$ for all $g \in G$. Thus $M(V)$ is an invariant subspace of $\tilde{U}$. Since $\tilde{U}$ is an irrep, we have either $M(V) = \tilde{V}$ or $M(V) = 0$. The former is impossible for dimensional reasons and thus $M(v) = 0$ for all $v \in V$, meaning $M = 0$.

In case 2, assume $\dim V > \dim \tilde{V}$. Here we can reason similarly that $W := \text{kernel}(M) = \{v \in V|M(v) = 0\}$ is an invariant subspace of U, and because it is an irrep conclude that $W = V$ and thus $M = 0$.

In case 3, if $\dim V = \dim \tilde{V}$, if either $M(V) = \tilde{V}$ or $W = V$, we find $M = 0$ by the same logic as in the previous cases. Thus, the remaining case for which we have not yet concluded $M = 0$ is $M(V) = \tilde{V}$ and $\text{kernel}(M) = 0$. This would however imply that M is both surjective and injective and thus a vector space isomorphism. Since it is also equivariant, it would witness an equivalence $U \cong \tilde{U}$, which contradicts the assumption. □

In other words, there are no intertwiners (linear M with the above property) between inequivalent irreps. Even between equivalent irreps, the intertwiners are constrained as follows.

Schur's lemma part 2: Let U an irrep of G on a vector space V over an algebraically closed field (e.g. over $\mathbb{C}$), and $M : V \to V$ equivariant. Then $M = \lambda\, \mathrm{id}_V$ for some scalar $\lambda \in \mathbb{C}$.

Proof: Since the field is algebraically closed, M has at least one eigenvalue λ. Let $0 \neq v \in V$ be a corresponding eigenvector. Now consider the eigenspace $E_\lambda = \{v \in V|M(v) = \lambda v\}$. It is an invariant subspace since for any $v \in E_\lambda$ and $g \in G$ we have $M(U(g)(v)) = U(g)(M(v)) = U(g)(\lambda v) = \lambda U(g)(v)$ and thus $U(g)(v) \in E_\lambda$. Since U is an irrep, we have either $E_\lambda = 0$ or $E_\lambda = V$. Since there is an eigenvector $v \neq 0$, the former is ruled out and we find $M(v) = \lambda v$ for all $v \in V$, or equivalently

$M = \lambda \mathrm{id}_V$. □

We find as a corollary that irreps of an abelian group G are one dimensional, if the underlying field is algebraically closed.

Proof: Let U be an irrep on a vector space V. Since G is abelian, we have $U(g)U(h) = U(gh) = U(hg) = U(h)U(g)$ for all $g, h \in G$. Thus Schurs Lemma part 2 applies with $M = U(h)$ and we find $U(g) = \lambda \mathrm{id}_V$ for some scalar λ, which may depend on g. This means that any subspace of V is invariant under U, which contradicts irreducibility of U unless there are no non-trivial proper subspaces, i.e. unless $\dim V = 1$. □

To exploit Schur's Lemma, it is convenient to use a basis $\{e_i\}$ for every vector space such that the group representation is not only equivalent but actually equal to a direct sum of irreps. This is done by employing the equivariant isomorphism – a basis change – guaranteed by the equivalence to a direct sum of irreps. For abelian groups, where all irreps are one-dimensional, this effectively assigns an irrep label $\mathbf{a}_i$ to every basis element e_i.

This imposes a sparsity structure on the matrix representations of equivariant maps, as follows. Let $f : V \to \tilde{V}$ be equivariant between representations $U = \bigoplus_{i=1}^{M} U_i$ on V and $\tilde{U} = \bigoplus_{j=1}^{N} \tilde{U}_j$ on $\tilde{V}$ of an *abelian* symmetry group G, where $U_i, \tilde{U}_j$ are irreps and $M = \dim V$, as well as $N = \dim \tilde{V}$. For the matrix elements $f_{ij} = \langle \tilde{e}_i | f(e_j) \rangle$ in the computational bases $\{e_j\}$ of V and $\{\tilde{e}_i\}$ of $\tilde{V}$, equivariance $f = \tilde{U}^\dagger(g) \circ f \circ U(g)$ translates to

$$f_{ij} = \left\langle \tilde{U}(g)(\tilde{e}_i) \middle| f(U(g)(e_j)) \right\rangle = \left\langle \tilde{U}_i(g)\tilde{e}_i \middle| f(U_j(g)e_j) \right\rangle = \tilde{U}_i^\dagger(g) f_{ij} U_j(g), \qquad \text{(A.5)}$$

where we used that $\tilde{U}_i(g)$ and $U_j(g)$ are just numbers and that f is linear. Thus for fixed indices $i = 1, \ldots N$ and $j = 1, \ldots, M$ the entry f_{ij} of the matrix representation is an equivariant map $\mathbb{C} \to \mathbb{C}$ between U_i and $\tilde{U}_j$ and by Schurs lemma part 1, we have $f_{ij} = 0$ if $U_i \ncong \tilde{U}_j$ are inequivalent. Thus, if we sort the basis elements by sector, that is by equivalence class of the irreps, we find a block diagonal structure for the matrix representation of f, and the allowed blocks are between basis elements $e_i \in V$ and $e_j \in W$ such that the irreps $U_i \cong \tilde{U}_j$ are in the same sector.

To apply the same idea to tensors, we need to define the product representation on the tensor product of vector spaces. The tensor product $U_1 \otimes U_2$ of representations U_i on vector spaces V_i is a representation on the tensor product $V_1 \otimes V_2$ which is given by $(U_1 \otimes U_2)(g) = U_1(g) \otimes U_2(g)$. Note that for abelian groups, the product of irreps, which are one-dimensional, is also one-dimensional and thus equivalent to a single irrep. This defines fusion rules of sectors, which we write as $\mathbf{a} + \mathbf{b} = \mathbf{c}$ if $U_\mathbf{a} \otimes U_\mathbf{b} \cong U_\mathbf{c}$. For the group $G = \mathrm{U}(1)$, for example, the sector labels are integers $\mathbf{a} \in \mathbb{Z}$ and the fusion rules are regular addition of labels. For a $G = \mathbb{Z}_N$ group, the sector labels are from $\mathbf{a} \in \mathbb{Z}_N$ and the fusion rules are addition modulo N, as the notation suggests. For products of groups, the fusion rules are componentwise.

The final ingredient is the natural representation on the dual space $V^\star$ – the space of bra vectors – given a representation on a "ket space" V. Given a representation

U of a group G on a space V with components $|\psi\rangle \mapsto U(g)|\psi\rangle$, the contragradient representation $\bar{U}$ is a representation on the dual space $V^\star$ given by $\bar{U}(g) = U(g^{-1})^{\mathrm{T}}$, which is the map $\langle\phi| \mapsto \langle\phi|U^\dagger(g)$. As a result, acting on both a bra and a ket vector with the respective representation leaves the inner products $(\langle\phi|U^\dagger(g))(U(g)|\psi\rangle) = \langle\phi|\psi\rangle$ invariant.

For abelian groups, the contragradient representation of an irrep, and in particular of a sector $\mathbf{a}$ is again one-dimensional, thus irreducible and equivalent to a single sector. This gives rise to the notion of the "opposite" sector and we write $-\mathbf{a}$ for that sector label such that $\bar{U}_{\mathbf{a}} \cong U_{-\mathbf{a}}$. We have an equivalence $\bar{U}_{\mathbf{a}} \otimes U_{\mathbf{a}} \cong U_{\mathbf{0}}$ to the trivial sector, where the isomorphism is given by linear extension of $S : \langle\phi| \otimes |\psi\rangle \mapsto \langle\phi|\psi\rangle$. In particular, this establishes $\mathbf{a} + (-\mathbf{a}) = \mathbf{0}$, justifying the notation for the opposite sector. For our examples U(1) ($\mathbb{Z}_N$), $-\mathbf{a}$ is actually the negative integer of $\mathbf{a}$ (modulo N), and for product groups it is componentwise.

A.2 Cyclic Groups $\mathbb{Z}_N$

In the following section, here for $G = \mathbb{Z}_N$, we give relevant properties, and in particular the topological data for a number of symmetries relevant in condensed matter physics.

The group $\mathbb{Z}_N$ for an integer $N > 1$ is given by the numbers $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$ with addition modulo N. Since it is an abelian group, all of its irreps are one-dimensional and there are N equivalence classes of irreps. For each $a = 0, \ldots, N-1$, we choose the following representative irrep for the sector a; $U_a(g) : \mathbb{C} \to \mathbb{C}, z \mapsto \mathrm{e}^{2\pi\mathrm{i}\frac{a}{N}g} z$, where $g \in \mathbb{Z}_N$. Thus, sectors are labeled by non-negative integers $a \in \mathcal{S} = \{0, 1, \ldots, N-1\}$. The dual sector is $\overline{a} = (-a \mod N)$. The fusion rules are

$$a \otimes b \cong (a + b \mod N), \tag{A.6}$$

where in the following we will drop explicitly writing $\mod N$ and all arithmetic is implicitly $\mod N$.

The N symbol is given by $N_c^{ab} = \delta_{a+b,c}$. For abelian groups, a fusion tree is fully determined by the uncoupled sectors and in the following we only give the values of the symbols for valid fusion channels.

The F symbol is trivial

$$\left[F^{abc}_{a+b+c}\right]^{b+c,1,1}_{a+b,1,1} = 1 \tag{A.7}$$

where we directly plugged in the only valid sectors, e.g. $d = a+b+c$ and multiplicity labels $\mu = \nu = \kappa = \lambda = 1$. The R symbol is similarly trivial

$$\left[R^{ab}_{a+b}\right]^1_1 = 1. \tag{A.8}$$

The set of valid fusion outcomes consists of only a single sector $\mathcal{F}_{a,b} = \{(a + b \mod N)\}$. The C symbol, like the F symbol, is one for the only valid fusion channel

$$\left[C^{abc}_{a+b+c}\right]^{a+b,1,1}_{a+c,1,1} = 1\ . \tag{A.9}$$

The B symbol is similarly

$$\left[B^{ab}_{a+b}\right]^1_1 = 1 \; . \tag{A.10}$$

As for any abelian symmetry, all sectors are one-dimensional $d_a = 1$. As for any group symmetry, the twists $\Theta_a = 1$ are trivial. The Frobenius Schur indicators $\chi_a = +1$ are all positive.

The fusion tensors are trivial, given by linear extension of

$$X^{ab}_{c,1} : 1 \otimes 1 \mapsto 1 \tag{A.11}$$

and the Z isomorphisms are also just one, meaning $Z_a : \mathbb{C}^\star \ni z \mapsto z \in \mathbb{C}$.

A.3 The abelian group U(1)

The abelian group U(1) is the unit circle $\{z \in \mathbb{C} || z| = 1\}$ in the complex plane with multiplication as group operation. It has a countably infinite number of sectors, indexed by $a \in \mathbb{Z}$, where the representative irrep is given by

$$U_a(\mathrm{e}^{\mathrm{i}\phi}) : \mathbb{C} \to \mathbb{C}, z \mapsto \mathrm{e}^{\mathrm{i}a\phi} z, \tag{A.12}$$

where $\mathrm{e}^{\mathrm{i}\phi} \in \mathrm{U}(1)$ is a general group element. All results for $\mathbb{Z}_N$ groups listed above also hold for U(1), if we replace the addition modulo N with regular addition.

A.4 The non-abelian group SU(2)

The compact Lie group SU(2) is given by complex 2×2 matrices which are unitary and are special (have determinant one). It has a countably infinite number of sectors, labeled by a half-integer "total spin" $j \in \frac{1}{2}\mathbb{Z}$.

For derivations and for concrete expressions of the representation, it is useful to go to the associated Lie algebra $\mathfrak{su}(2)$, and later to its complexification. A general element $g \in \mathrm{SU}(2)$ can always be written as e^{ℓ} for some $\ell \in \mathfrak{su}(2)$. For the spin j irrep $U_j : \mathrm{SU}(2) \mapsto \mathrm{Hom}\left(R_j,\ R_j\right)$ of SU(2), there is a compatible representation $\pi_j : \mathfrak{su}(2) \to \mathrm{Hom}\left(R_j,\ R_j\right)$ on the same representation space $R_j = \mathbb{C}^{2j+1}$, such that $U_j(g) = \exp(\pi_j(\ell))$. This allows us to build the group irreps from the Lie algebra representations, which are easier to deal with due to the vector space structure. The standard $\mathbb{C}$-basis for the complexified Lie algebra $\mathfrak{su}(2)_\mathbb{C}$ consists of three elements j_3, j_+, j_-. Their representations $J_k = \pi_j(j_k)$ are the following operators, given as $(2j+1) \times (2j+1)$ matrices

$$
\begin{gathered}
J_3 = \begin{pmatrix} j & & & & \\ & j-1 & & & \\ & & \ddots & & \\ & & & 1-j & \\ & & & & -j \end{pmatrix} \\
J_+ = \begin{pmatrix} 0 & & & & \\ 1 & 0 & & & \\ & 1 & \ddots & & \\ & & \ddots & 0 & \\ & & & 1 & 0 \end{pmatrix} \quad J_+ = \begin{pmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ & & & & 0 \end{pmatrix} .
\end{gathered}
\tag{A.13}
$$

They are related to the standard $\mathbb{R}$-basis of the real Lie algebra $\mathfrak{su}(2)$ as $J_3 = \mathrm{i}L_3$ and $J_\pm = \mathrm{i}L_1 \mp L_2$. Or conversely $L_1 = -\frac{\mathrm{i}}{2}(J_+ + J_-)$, $L_2 = -\frac{1}{2}(J_+ - J_-)$ and $L_3 = -\mathrm{i}J_3$. Note that the "real" qualifier of the real Lie algebra refers to the fact that $\mathfrak{su}(2)$ is a real vectorspace, and that linear combinations of its basis elements need to have real coefficients. The representation matrices L_i, however, have complex entries and the representation space R_j is a complex vector space.

The $j = 0$ case gives us the trivial representation $U_{j=0}(g) : z \mapsto z$. The $j = 1/2$ case gives us the faithful representation $U(g) = g$.

The fusion rules are $j_1 \otimes j_2 = \bigoplus_{J=|j_1-j_2|}^{j_1+j_2} J$, such that

$$
N_c^{ab} = \begin{cases} 1 & c \in \{|a-b|, \dots, a+b\} \\ 0 & \text{else} \end{cases} . \tag{A.14}
$$

We can directly read off the set of fusion outcomes $\mathcal{F}_{a,b} = \{|a-b|, \dots, a+b\}$. Note that if a, b are either both integer or both fractional, the fusion outcomes are all integer. Conversely if either a or b, but not both, are fractional, all fusion outcomes are fractional. Since the N symbol can not take values greater than one, the only possible multiplicity label on a valid fusion tensor is $\mu = 1$.

For the fusion tensors, we choose the usual Clebsch-Gordan coefficients, commonly denoted as $\langle j_1 m_1 j_2 m_2 | JM \rangle$. In the standard z-basis $\{|m\rangle | m = -j, \dots, j\}$ for the representation space R_j of the spin-j irrep, that means

$$
X_{J,1}^{j_1,j_2} = |m_1\rangle \otimes |m_2\rangle \mapsto \sum_{M=-J}^{J} \langle JM | j_1 m_1 j_2 m_2 \rangle \, |M\rangle \ . \tag{A.15}
$$

The F symbol, as we define it in (5.70), is related to the Wigner 6j symbol or the Racah W symbol $W(j_1 j_2 J j_3; J_{12} J_{23})$ as follows

$$
\begin{aligned}
\left[F_d^{abc}\right]_{f,1,1}^{e,1,1} &= \sqrt{2e+1}\sqrt{2f+1} W(abdc; fe) \\
&= \sqrt{2e+1}\sqrt{2f+1}(-1)^{a+b+c+d} \begin{Bmatrix} a & b & f \\ c & d & e \end{Bmatrix} .
\end{aligned}
\tag{A.16}
$$

For computation and storage schemes for the 6j symbols which give the F symbols and the Wigner 3j symbols, which give the fusion tensors refer e.g. to reference [175].

The R symbol is given by

$$\left[R^{ab}_{c}\right]^{1}_{1} = (-1)^{a+b-c} \ . \tag{A.17}$$

Note that the exponent is integer for valid fusion channels $N^{ab}_c > 0$.

The quantum dimensions are $d_a = 2a+1$. The Frobenius Schur indicator is given by$\chi_a = (-1)^{(2a)}$. It is positive for integer spins and negative for half integers, while the twist, like for all group symmetries, is $\Theta_a = +1$.

The Z isomorphism takes the following explicit form in the standard z-basis $\{|m\rangle\}$.

$$Z_j : R_j^\star \to R_j, \langle m| \mapsto \sum_n A^j_{m,n} |n\rangle \ , \tag{A.18}$$

where

$$A^j = \begin{pmatrix} & & & & -\chi_j \\ & & & \chi_j & \\ & & \iddots & & \\ & -1 & & & \\ 1 & & & & \\ -1 & & & & \end{pmatrix} \tag{A.19}$$

is a $(2j+1) \times (2j+1)$ anti-diagonal matrix with alternating signs, such that its top right entry is $-\chi_j = (-1)^{2j+1}$. We have chosen the phase of the Z ismorphism, such that (5.84) holds with the usual phase choice of the Clebsch Gordan coefficients.

A.4.1 Derivation of the Z isomorphism

For completeness, and to facilitate similar treatment of other Lie groups, we give the full derivation of the above result for the Z isomorphism in the following. First, as a small warning, note that Z_j is a *linear* map from bra vectors to ket vectors. This is rather unusual in physics, where we may have a strong expectation that such maps, such as e.g. the dagger, are anti-linear.

We show that the map defined above has the necessary properties to be a Z isomorphism. First, it is by definition linear and inherits unitarity from the matrix A. It remains to check that Z_j is equivariant.

It is straight-forward to check that $(A^j)^\dagger J_i A^j = -J_i$ for the basis $J_3, J_\pm$ of the complexified Lie algebra. This implies $(A^j)^\dagger L_i A^j = \bar{L}_i$ for the basis of the real Lie algebra, where $\bar{L}$ denotes the elementwise complex conjugate matrix of L.

The representation $\sum_i \alpha_i L_i = L = \pi_j(l)$ of a general element $\mathfrak{su}(2) \ni l = \sum_i \alpha_i l_i$ is a real ($\alpha_i \in \mathbb{R}$) linear combination of the L_i and thus also fulfills $(A^j)^\dagger L A^j = \bar{L}$. Therefore the respective representation $U(g)$ of a general element $SU(2) \ni g = \mathrm{e}^l$ fulfills

$$(A^j)^\dagger U(g) A^j = (A^j)^\dagger \mathrm{e}^L A^j = \mathrm{e}^{(A^j)^\dagger L A^j} = \mathrm{e}^{\bar{L}} = \bar{U}(g) \ . \tag{A.20}$$

This is sufficient to show that Z_j is symmetry-preserving, meaning $Z_j \circ U_{(R_j)^\star}(g) = U_{R_j}(g) \circ Z_j$ for all $g \in SU(2)$. It is enough to compare the action on a generic basis element;

$$\begin{aligned}
\text{LHS: } \langle m| &\mapsto \langle m| U^\dagger(g) = \sum_{m'} \overline{U}_{m'm} \langle m'| \mapsto \sum_{m'n} \overline{U}_{m'm} A^j_{nm'} |n\rangle \\
&= \sum_n \left[A^j \cdot \overline{U}(g)\right]_{nm} |n\rangle
\end{aligned} \tag{A.21}$$

$$\begin{aligned}
\text{RHS: } \langle m| &\mapsto \sum_{m'} A^j_{m'm} |m'\rangle \mapsto \sum_{m'} A^j_{m'm} U(g) |m'\rangle = \sum_{m'n} A^j_{m'm} U_{nm'}(g) |n\rangle \\
&= \sum_n \left[U(g) A^j\right]_{nm} |n\rangle = \sum_n \left[A^j (A^j)^\dagger U(g) A^j\right]_{nm} |n\rangle \\
&= \sum_n \left[A^j \overline{U}(g)\right]_{nm} |n\rangle \ ,
\end{aligned} \tag{A.22}$$

where we write $U(g) := U_{R_j}(g)$ for the representation on the ket space R_j for readability and $U_{nm} := \langle n|U(g)|m\rangle$.

To derive the concrete form of the matrix in general, note that we used above that the irreps of SU(2) are self-dual, meaning $\overline{j} = j$. In general, we need to solve the set of matrix equations $(A^j)^\dagger \pi_j(\ell_i) A^j = \overline{\pi}_{\overline{j}}(\ell_i)$ for all generators ℓ_i of the real Lie algebra, as the remaining derivations generalizes to any Lie group. Note that depending in the chosen bases, this may or may not translate to an equation like $(A^j)^\dagger \pi_j(j_i) A^j = -\pi_{\overline{j}}(j_i)$ with the generators j_i of the complexified algebra.

A.5 Fermions

Next, we consider the tensor category **Ferm**. We can understand it as a description of fermions since the fusion rules and the exchange statistics encoded in the braid give the correct behavior.

We start from the category $\mathbf{FdSHilb}_{\mathbb{C}}$ of finite dimensional complex super Hilbert spaces. Its objects are pairs (H, H') of finite dimensional complex Hilbert spaces, where we can think of H as the "bosonic" part with even fermionic parity and H' as the "fermionic" part with odd parity. Its morphisms are pairs $(f, f') : (H, H') \to (K, K')$ of linear maps $f : H \to K$ and $f' : H' \to K'$ and composition is componentwise, as is addition of morphisms and multiplication with scalars. The identity morphism is $\mathrm{id}_{(H,H')} = (\mathrm{id}_H, \mathrm{id}_{H'})$. Direct sums are componentwise direct sums of vector spaces. We obtain the tensor category **Ferm** by equipping the super Hilbert spaces with the following monoidal and braiding structures.

For explicit constructions in the following, it is convenient to use the matrix notation in a category with linear structure and direct sums. Let $A = \oplus_{m=1}^M A_m$ be witnessed by inclusions $i_m : A_m \to A$ and projections $p_m : A \to A_m$ and similarly $B = \oplus_{n=1}^N B_n$ by inclusions $\tilde{i}_n : B_n \to B$ and projections $\tilde{p}_n : B \to B_m$. Now for morphisms

$f_{m,n} : A_m \to B_n$ we define their matrix as

$$\begin{pmatrix} f_{1,1} & f_{2,1} & \cdots & f_{M,1} \\ f_{1,2} & f_{2,2} & \cdots & f_{M,2} \\ \vdots & \vdots & \ddots & \vdots \\ f_{1,N} & f_{2,N} & \cdots & f_{M,N} \end{pmatrix} := \sum_{m,n} \tilde{i}_n \circ f_{m,n} \circ p_m \tag{A.23}$$

and conversely, every morphism $g : A \to B$ is equal to the matrix of $g_{m,n} := \tilde{p}_n \circ g \circ i_m$. If the morphisms are linear maps between vector spaces, we can think of this matrix notation as forming the block matrix of the respective matrix representations of the $f_{m,n}$. Composition of matrix morphisms can be carried out similar to matrix-matrix multiplication, i.e. in short $(f \circ g)_{m,n} = \sum_k f_{m,k} \circ g_{k,n}$.

The tensor product is defined on objects as

$$(H, H') \otimes (K, K') := ((H \otimes K) \oplus (H' \otimes K'), (H \otimes K') \oplus (H' \otimes K)) \tag{A.24}$$

and encodes the fermionic statistics; a composite system of two fermionic degrees of freedom $(0, H')$ and $(0, K')$ behaves as a boson $(H' \otimes K', 0)$. The tensor product of morphisms $(f, f') : (H, H') \to (K, K')$ and $(g, g') : (L, L') \to (M, M')$ is defined as

$$(f, f') \otimes (g, g') := \left(\begin{pmatrix} f \otimes g & 0 \\ 0 & f' \otimes g' \end{pmatrix}, \begin{pmatrix} f \otimes g' & 0 \\ 0 & f' \otimes g \end{pmatrix} \right), \tag{A.25}$$

structurally very similar to the tensor product of objects, except diagonal matrices of morphisms take the place of direct sums of spaces. To read explicit forms of morphisms like above, first expand the domain and codomain of $(f, f') \otimes (g, g') : (H, H') \otimes (L, L') \to (K, K') \otimes (M, M')$ using (A.24). Now, each component of such a morphism is a map between the direct sums, resulting from expanding the tensor product, and can be given as a matrix.

The monoidal unit is $I = (\mathbb{C}, 0)$. The associator is given by

$$\alpha_{(A,A')(B,B')(C,C')} = \left(\begin{pmatrix} \alpha_{ABC} & 0 & 0 & 0 \\ 0 & 0 & \alpha_{AB'C'} & 0 \\ 0 & 0 & 0 & \alpha_{A'BC'} \\ 0 & \alpha_{A'B'C} & 0 & 0 \end{pmatrix}, \begin{pmatrix} \alpha_{ABC'} & 0 & 0 & 0 \\ 0 & 0 & \alpha_{AB'C} & 0 \\ 0 & 0 & 0 & \alpha_{A'BC} \\ 0 & \alpha_{A'B'C'} & 0 & 0 \end{pmatrix} \right), \tag{A.26}$$

where $\alpha_{ABC} : (A \otimes B) \otimes C \xrightarrow{\cong} A \otimes (B \otimes C)$ are the associators of $\mathbf{FdHilb}_{\mathbb{C}}$. The definition in terms of matrices suppresses some isomorphisms of the form $(H \oplus K) \otimes L \xrightarrow{\cong} (H \otimes L) \oplus (K \otimes L)$ to map the domain $((A, A') \otimes (B, B')) \otimes (C, C')$ such that each component is a flat direct sum and we can apply the matrix notation.

The unitors are

$$\lambda_{(A,A')} = \left(\begin{pmatrix} \lambda_A & 0 \end{pmatrix}, \begin{pmatrix} \lambda_{A'} & 0 \end{pmatrix} \right) \qquad \rho_{(A,A')} = \left(\begin{pmatrix} \rho_A & 0 \end{pmatrix}, \begin{pmatrix} 0 & \rho_{A'} \end{pmatrix} \right). \tag{A.27}$$

The dagger of morphisms is componentwise $(f, g)^\dagger = (f^\dagger, g^\dagger)$.

The dual is also componentwise, $(H, K)^\star = (H^\star, K^\star)$, where the cup and cap are given by

$$\eta_{(H,H')} = \left(\begin{pmatrix}\eta_H \\ \eta_{H'}\end{pmatrix}, 0\right) \qquad \epsilon_{(H,H')} = \left(\begin{pmatrix}\epsilon_H & \epsilon_{H'}\end{pmatrix}, 0\right) \ . \tag{A.28}$$

The braid is given by

$$\tau_{(A,A'),(B,B')} = \left(\begin{pmatrix}\tau_{A,B} & 0 \\ 0 & -\tau_{A',B'}\end{pmatrix}, \begin{pmatrix}\tau_{A,B'} & 0 \\ 0 & \tau_{A',B}\end{pmatrix}\right), \tag{A.29}$$

where $\tau_{A,B} : A \otimes B \to B \otimes A, |a\rangle \otimes |b\rangle \mapsto |b\rangle \otimes |a\rangle$ is the braid of $\mathbf{FdHilb}_\mathbb{C}$. We see the fermionic exchange statistics; braiding a fermionic state around a fermionic state gives a minus sign, while all other braidings have a plus sign. The braid is symmetric.

This fully defines the category and we now turn to the topological data. Checking the compatibility axioms to verify that the definitions above do indeed yield a tensor category is cumbersome, but straight-forward, as is deriving the topological data below.

There are two sectors, the trivial sector, or boson $I = (\mathbb{C}, 0)$ and the fermion $\psi = (0, \mathbb{C})$, such that $\mathcal{S} = \{I, \psi\} = \mathbb{Z}_2$, where we identify $I = 0 \in \mathbb{Z}_2$ and $\psi = 1 \in \mathbb{Z}_2$ for explicit numerical values below. Both sectors are self-dual; $\bar{\mathbf{a}} = \mathbf{a}$. The fusion rules resulting from the tensor product defined above are $I \otimes I \cong I \cong \psi \otimes \psi$ and $I \otimes \psi \cong \psi \cong \psi \otimes I$. This category shares a property with representations of abelian groups, which we call *unique fusion*, namely that the tensor product of two sectors a, b decomposes as only a single sector, such that N_c^{ab} is non-zero for only a single sector c, where it is one. We write $a+b$ for that single sector such that $a \otimes b \cong a+b$. We find

$$N_c^{ab} = \delta_{c,a+b}. \tag{A.30}$$

Since we have unique fusion, the entire fusion channel is determined by the input sectors, i.e. the upper indices of the F, R, C, B symbols. We therefore only give the values of the symbols with the unique indices that give a consistent fusion channel.

The fusion tensors are

$$\begin{aligned} X_I^{II} &= \left(\begin{pmatrix}X \\ 0\end{pmatrix}, 0\right) & X_\psi^{I\psi} &= \left(0, \begin{pmatrix}X \\ 0\end{pmatrix}\right) \\ X_\psi^{\psi I} &= \left(0, \begin{pmatrix}0 \\ X\end{pmatrix}\right) & X_I^{\psi\psi} &= \left(\begin{pmatrix}0 \\ X\end{pmatrix}, 0\right), \end{aligned} \tag{A.31}$$

where $X : \mathbb{C} \otimes \mathbb{C} \to \mathbb{C}, x \otimes y \to xy$ is the fusion tensor of $\mathbf{FdHilb}_\mathbb{C}$.

The Z isomorphism are simply $Z_I = (Z, 0)$ and $Z_\psi = (0, Z)$, where $Z : \mathbb{C}^\star \to \mathbb{C}, z \mapsto z$ is the Z isomorphism of $\mathbf{FdHilb}_\mathbb{C}$.

As for all categories with unique fusion, we find that the F symbol for valid fusion channels is

$$\left[F_{a+b+c}^{abc}\right]_{a+b,1,1}^{b+c,1,1} = 1. \tag{A.32}$$

The R symbol is given by

$$\left[R^{ab}_{a+b}\right]^{1}_{1} = 1 - 2ab = \begin{cases} -1 & a = b = \psi \\ 1 & \text{else} \end{cases} \tag{A.33}$$

as expected; if we braid two fermions, we get a minus sign, and a plus sign otherwise.

The resulting expression for the C symbol is

$$\left[C^{abc}_{a+b+c}\right]^{a+b,1,1}_{a+c,1,1} = 1 - 2bc = \begin{cases} -1 & b = c = \psi \\ 1 & \text{else} \end{cases} \tag{A.34}$$

and the B symbol is again trivial

$$\left[B^{ab}_{a+b}\right]^{1}_{1} = 1. \tag{A.35}$$

Both sectors are one-dimensional $d_a = 1$ and have a positive Frobenius indicators $\chi_a = +1$. The twist is $\Theta_a = (-1)^a = 1 - 2a$. In that sense, we have equipped $\mathbf{FdSHilb}_{\mathbb{C}}$ with a non-trivial twist.

A.6 Fibonacci Anyons

Next, we consider the tensor category **Fib**. We can understand it as a description of Fibonacci anyon excitations since the fusion rules and the exchange statistics encoded in the braid give the correct behavior.

The construction is similar to the category **Ferm** of fermions and comments there apply. We start from the same underlying objects, tuples (H, H') of finite dimensional Hilbert spaces, and morphisms, tuples $(f, f') : (H, H') \to (K, K')$ of linear maps $f : H \to K$ and $f' : H' \to K'$. Again, composition, addition, scalar multiplication, identities, direct sums and the dagger are all componentwise.

We choose a different tensor product for **Fib**, however, namely

$$(H, H') \otimes (K, K') := ((H \otimes K) \oplus (H' \otimes K'), (H \otimes K') \oplus (H' \otimes K) \oplus (H' \otimes K')) \,. \tag{A.36}$$

Note the additional term in the second component compared to (A.24). The monoidal unit is $I = (\mathbb{C}, 0)$. The associator

$$\alpha_{(A,A')(B,B')(C,C')} = \left(\alpha^{I}_{(A,A')(B,B')(C,C')}, \alpha^{\tau}_{(A,A')(B,B')(C,C')}\right)$$

is a tuple of the following two components

$$\alpha^{I}_{(A,A')(B,B')(C,C')} = \begin{pmatrix} \alpha_{ABC} & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_{AB'C'} & 0 & 0 \\ 0 & 0 & 0 & \alpha_{A'BC'} & 0 \\ 0 & \alpha_{A'B'C} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha_{A'B'C'} \end{pmatrix}$$

$$\alpha^{\tau}_{(A,A')(B,B')(C,C')} = \begin{pmatrix} \alpha_{ABC'} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_{AB'C} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha_{AB'C'} & 0 & 0 \\ 0 & 0 & 0 & \alpha_{A'BC} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\phi}\alpha_{A'B'C'} & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{\phi}}\alpha_{A'B'C'} \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_{A'BC'} & 0 \\ 0 & 0 & 0 & 0 & \alpha_{A'B'C} & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{\phi}}\alpha_{A'B'C'} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\phi}\alpha_{A'B'C'} \end{pmatrix}, \tag{A.37}$$

where $\phi = (1+\sqrt{5})/2$ is the golden ratio and we suppress isomorphisms similar to (A.26).

The unitors are

$$\lambda_{(A,A')} = \left(\begin{pmatrix}\lambda_A & 0\end{pmatrix}, \begin{pmatrix}\lambda_{A'} & 0 & 0\end{pmatrix}\right) \qquad \rho_{(A,A')} = \left(\begin{pmatrix}\rho_A & 0\end{pmatrix}, \begin{pmatrix}0 & \rho_{A'} & 0\end{pmatrix}\right). \tag{A.38}$$

The dual object is componentwise $(H,K)^\star = (H^\star, K^\star)$, where the cup and cap are given by

$$\eta_{(H,H')} = \left(\begin{pmatrix}\eta_H \\ \eta_{H'}\end{pmatrix}, 0\right) \qquad \epsilon_{(H,H')} = \left(\begin{pmatrix}\epsilon_H & \epsilon_{H'}\end{pmatrix}, 0\right) \tag{A.39}$$

which is structurally very similar to fermions, but note that the zero maps in the second components have a different type.

The braid is given by

$$\tau_{(A,A'),(B,B')} = \left(\begin{pmatrix}\tau_{A,B} & 0 \\ 0 & e^{-\frac{4}{5}\pi i}\tau_{A',B'}\end{pmatrix}, \begin{pmatrix}\tau_{A,B'} & 0 & 0 \\ 0 & \tau_{A',B} & 0 \\ 0 & 0 & e^{\frac{3}{5}\pi i}\tau_{A',B'}\end{pmatrix}\right) \tag{A.40}$$

Here, we can already heuristically interpret the exchange statistics. Braiding two τ's results in a phase that depends on the joint state or "fusion channel", it is $e^{-\frac{4}{5}\pi i}\tau_{A',B'}$ if they fuse to I and $e^{\frac{3}{5}\pi i}$ if they fuse to τ. In particular, the braid is *not* symmetric.

Now for the topological data; The sectors are the trivial sector or vacuum $I = (\mathbb{C}, 0)$ and the tau anyon $\tau = (0, \mathbb{C})$, such that $\mathcal{S} = \{I, \tau\}$. They are both self-dual $\overline{a} = a$ and the fusion rules

$$I \otimes I \cong I \quad I \otimes \tau \cong \tau \quad \tau \otimes I \cong \tau \quad \tau \otimes \tau \cong I \oplus \tau \tag{A.41}$$

are indeed the correct fusion rules for Fibonacci anyons and give rise to the N symbol

$$N_c^{Ib} = \delta_{b,c} \qquad N_c^{aI} = \delta_{a,c} \qquad N_c^{\tau\tau} = 1. \tag{A.42}$$

Note that the N symbol only takes on values $N_c^{ab} \in \{0, 1\}$, such that all multiplicity labels for valid fusion channels are $\mu = 1$.

The fusion tensors are

$$\begin{gathered} X_I^{II} = \left(\begin{pmatrix} X \\ 0 \end{pmatrix}, 0\right) \qquad X_\tau^{I\tau} = \left(0, \begin{pmatrix} X \\ 0 \\ 0 \end{pmatrix}\right) \qquad X_\tau^{\tau I} = \left(0, \begin{pmatrix} 0 \\ X \\ 0 \end{pmatrix}\right) \\ X_I^{\tau\tau} = \left(\begin{pmatrix} 0 \\ X \end{pmatrix}, 0\right) \qquad X_\tau^{\tau\tau} = \left(0, \begin{pmatrix} 0 \\ 0 \\ X \end{pmatrix}\right). \end{gathered} \tag{A.43}$$

The Z isomorphism are simply $Z_I = (Z, 0)$ and $Z_\tau = (0, Z)$.

The only non-trivial F symbol is

$$\left[F_\tau^{\tau\tau\tau}\right]_{f,1,1}^{e,1,1} = \begin{cases} \phi^{-1} & e = f = I \\ -\phi^{-1} & e = f = \tau \\ \phi^{-1/2} & (e, f) \in \{(I, \tau), (\tau, I)\} \end{cases} \tag{A.44}$$

and for all other $(a, b, c, d) \neq (\tau, \tau, \tau, \tau)$ we have $[F_d^{abc}]_{f,1,1}^{e,1,1} = 1$ where there is only one valid choice for e, f in each case.

The only non-trivial R symbol is

$$\left[R_c^{\tau\tau}\right]_1^1 = \begin{cases} \mathrm{e}^{-\frac{4}{5}\pi\mathrm{i}} & c = I \\ \mathrm{e}^{\frac{3}{5}\pi\mathrm{i}} & c = \tau \end{cases} \tag{A.45}$$

and for all other $(a, b) \neq (\tau, \tau)$ we have $[R_c^{ab}]_1^1 = 1$ and there is only one valid choice for c in each case.

The non-trivial B symbols are $[B_I^{\tau\tau}]_1^1 = \phi^{-1/2}$ and $[B_\tau^{I\tau}]_1^1 = \phi^{1/2}$ and all others are $[B_\tau^{\tau\tau}]_1^1 = [B_\tau^{\tau I}]_1^1 = [B_I^{II}]_1^1 = 1$.

The quantum dimensions are $d_I = 1$ and $d_\tau = \phi$, the Frobenius Schur indicators are all positive $\chi_a = +1$ and the twists are $\theta_I = 1$ and $\theta_\tau = \phi^{-1}\mathrm{e}^{-\frac{4}{5}\pi\mathrm{i}} + \mathrm{e}^{\frac{3}{5}\pi\mathrm{i}}$.

A.7 Combining symmetries

Combining multiple symmetries is formalized by Deligne's tensor product. For tensor categories $\mathbf{A}$ and $\mathbf{B}$, a Deligne tensor product is a category $\mathbf{A} \boxtimes \mathbf{B}$ with a functor $\boxtimes : \mathbf{A} \times \mathbf{B} \to \mathbf{A} \boxtimes \mathbf{B}$ that has the following defining property. Firstly, $\boxtimes$ is right-exact in both arguments and, secondly, for any other tensor category $\mathbf{C}$ and functor

$F : \mathbf{A} \times \mathbf{B} \to \mathbf{C}$ that is right exact in both arguments there is a unique right exact functor $\tilde{F} : \mathbf{A} \boxtimes \mathbf{B} \to \mathbf{C}$ such that $F = \tilde{F} \circ \boxtimes$. Under our assumptions for a tensor category, such a product is guaranteed to exist and is unique up to a unique equivalence [167, Prop 1.11.2] and can be equipped with all the structures of a tensor category in a canonical way [167, Prop 4.6.1]. Dealing with it explicitly and deriving the following statements rigorously is beyond the scope of this thesis. We expect that the explicit construction in [166, Def. 8.45] may prove useful. The following statements in this section are thus to be taken as conjecture.

We expect that the Deligne tensor product of tensor categories that describe individual symmetries describes the composite symmetry in the following senses; For symmetry groups G and H the Deligne product can be chosen as $\mathbf{FdRep}_G \boxtimes \mathbf{FdRep}_H = \mathbf{FdRep}_{G\times H}$ and thus describes the symmetry group $G \times H$. For a symmetry group G, we conjecture that the Deligne product $\mathbf{Ferm}_G := \mathbf{Ferm} \boxtimes \mathbf{FdRep}_G$ is a category that can be explicitly defined by generalizing the definition of **Ferm** from tuples of Hilbert spaces (objects in $\mathbf{FdHilb}_{\mathbb{C}}$) to tuples of representations of G, that is objects of $\mathbf{FdRep}_G$. It thus represents fermionic degrees of freedom with a symmetry group. Similarly for **Fib**, we expect $\mathbf{Fib}_G := \mathbf{Fib} \boxtimes \mathbf{FdRep}_G$ to describe Fibonacci anyon excitations with a symmetry group, with an analogous generalization of the definition of **Fib**. We expect that similar constructions apply to other modular anyon categories, other than **Fib**.

Let us now conjecture the topological data. The sectors are given by Deligne tensor products of sectors, that is

$$\mathcal{S}_{A\boxtimes B} = \{a \boxtimes b | a \in \mathcal{S}_A, b \in \mathcal{S}_B\} \tag{A.46}$$

which we can just understand as tuples (a, b) of respective sectors in the practical implementation. The topological data mostly just factorizes; The N symbol is given by

$$N^{a_1\boxtimes a_2, b_1\boxtimes b_2}_{c_1\boxtimes c_2} = N^{a_1,b_1}_{c_1} N^{a_2,b_2}_{c_2} \tag{A.47}$$

such that multiplicity labels $\mu = 1, \ldots, N^{a_1\boxtimes a_2, b_1\boxtimes b_2}_{c_1\boxtimes c_2}$ can be recast as tuples (μ_1, μ_2) of separate multiplicity labels $\mu_i = 1, \ldots N^{a_i,b_i}_{c_i}$, i.e. by using strides. The fusion tensors are

$$X^{a_1\boxtimes a_2, b_1\boxtimes b_2}_{c_1\boxtimes c_2,(\mu_1,\mu_2)} = X^{a_1,b_1}_{c_1,\mu_1} \boxtimes X^{a_2,b_2}_{c_2,\mu_2} \tag{A.48}$$

and the Z isomorphisms are

$$Z_{a_1\boxtimes a_2} = Z_{a_1} \boxtimes Z_{a_2} \ . \tag{A.49}$$

The remaining topological data all factorizes, such as e.g. for the R symbol

$$\left[R^{a_1\boxtimes a_2, b_1\boxtimes b_2}_{c_1\boxtimes c_2}\right]^{(\mu_1,\mu_2)}_{(\nu_1,\nu_2)} = \left[R^{a_1,b_1}_{c_1}\right]^{\mu_1}_{\nu_1} \left[R^{a_2,b_2}_{c_2}\right]^{\mu_2}_{\nu_2} \tag{A.50}$$

i.e. as a matrix of multiplicity indices, it is the Kronecker matrix product of the respective symbols from the separate categories. Similar expressions hold for the F, C and B symbols. The quantum dimensions, Frobenius Schur indicator and twist all factorize too, e.g. $d_{a_1\boxtimes a_2} = d_{a_1} d_{a_2}$.

Generalizations to combinations of more than two symmetries is straight-forward.

Appendix B

Derivation of automatic differentiation formulae

In this chapter, we give derivations for the results of AD formulae for the truncated SVD stated in section 4.2.2, and give similar formulae for a truncated hermitian eigendecomposition. This is inspired by the work in Ref. [39], where the authors analyze the AD formula for a truncated hermitian eigendecomposition and derive an additional term that arises from the truncated spectrum, as well as demonstrate how the AD formula simplifies for the particular cost function(s) they consider in the context of iPEPS optimization, because of an enlarged gauge invariance. The results that we present here for the general case are thus just a reiteration of what they found. The point we want to emphasize here is the simplification that arises in the presence of larger gauge freedoms of the decomposition. For the hermitian eigendecomposition, this is a reframing of the results of Francuz et al to a general context that does not rely on the specific usecase of the CTMRG step for which they derive it. For the SVD, this is – to our knowledge – a new result.

Let us first state some preliminary properties, used in the following derivations. The Hadamard product $A \circ B$ is defined as elementwise multiplication $(A \circ B)_{ij} := A_{ij} B_{ij}$. We can use two special matrices, the identity matrix $\mathbb{1}$ with entries $\mathbb{1}_{ij} = \delta_{i,j}$ and the fully off-diagonal matrix $\mathcal{O}$ with entries $\mathcal{O}_{ij} = 1 - \delta_{i,j}$, to decompose any matrix A into its diagonal and off-diagonal parts

$$A = \mathbb{1} \circ A + \mathcal{O} \circ A. \tag{B.1}$$

For a diagonal matrix S we have

$$\mathbb{1} \circ S = S \qquad , \qquad \mathcal{O} \circ S = 0. \tag{B.2}$$

and for diagonal S and a general matrix A we find commutative behavior on the diagonal

$$\mathbb{1} \circ (AS) = (\mathbb{1} \circ A) S = S(\mathbb{1} \circ A) = \mathbb{1} \circ (SA). \tag{B.3}$$

Conversely, the anticommutator

$$AS - SA = \mathcal{O} \circ (AS - SA) \tag{B.4}$$

is purely off-diagonal.

We also need the following properties of the Hadamard product for matrices A, B, C and diagonal S

$$(A \circ B)^{\mathrm{T}} = A^{\mathrm{T}} \circ B^{\mathrm{T}} \quad ; \quad (A \circ B)^\dagger = A^\dagger \circ B^\dagger \tag{B.5}$$

$$\mathrm{Tr}\,(S(\mathbb{1} \circ A)) = \mathrm{Tr}\,(SA) \tag{B.6}$$

$$\mathrm{Tr}\,(A(C \circ B)) = \mathrm{Tr}\left((C^{\mathrm{T}} \circ A)B\right) \ . \tag{B.7}$$

And note that the trace fulfills

$$\mathrm{Tr}\,(A + \text{c.c.}) = \mathrm{Tr}\left(A^\dagger + \text{c.c.}\right) \tag{B.8}$$

B.1 Derivations for hermitian eigendecomposition

The setup for the truncated hermitian eigendecompositition is a decomposition

$$A = USU^\dagger + XYX^\dagger \tag{B.9}$$

of a hermitian $n \times n$ matrix A, such that $U \in \mathbb{C}^{n \times k}$ and $X \in \mathbb{C}^{n \times (n-k)}$ are (left) isometries

$$U^\dagger U = \mathbb{1}_k \qquad ; \qquad X^\dagger X = \mathbb{1}_{n-k} \tag{B.10}$$

and X is the orthogonal complement of U

$$U^\dagger X = 0 = X^\dagger U \tag{B.11}$$

$$UU^\dagger + XX^\dagger = \mathbb{1}_n. \tag{B.12}$$

The matrix S of kept eigenvalues is real and diagonal

$$\mathbb{1} \circ S = S = S^\dagger. \tag{B.13}$$

We understand the decomposition as a function $A \mapsto (U, S)$ that achieves $A \approx USU^\dagger$.

B.1.1 Result

The result for the backward formula for the function $A \mapsto (U, S)$ is the following

$$\Gamma_{\overline{A}} = \Gamma^{\mathrm{S}}_{\overline{A}} + \Gamma^{\mathrm{U}\circ}_{\overline{A}} + \Gamma^{\mathrm{tr}}_{\overline{A}} = U\Gamma_{\overline{S}}U^\dagger + U\left(-F \circ \left(U^\dagger \Gamma_{\overline{U}}\right)\right)U^\dagger + XX^\dagger \gamma U^\dagger, \tag{B.14}$$

where

$$F_{ij} := \begin{cases} 0 & i = j \\ 1/(S_i - S_j) & i \neq j \end{cases} \tag{B.15}$$

and $\circ$ denotes elementwise multiplication of matrices, and where γ is the solution of the Sylvester equation

$$\Gamma_{\overline{U}} = \gamma S - A^\dagger XX^\dagger \gamma. \tag{B.16}$$

This Sylvester equation has a unique solution if and only if S and Y have disjoint spectra, such that splitting multiplets of degenerate eigenvalues should be avoided. In practice, the Sylvester equation should be solved numerically. Note that X is typically not computed and thus its projector should be applied to a matrix M via $XX^\dagger M = M - UU^\dagger M$. Francuz et al report encountering bad conditioning of the Sylvester equation and recommend a right pre-conditioner S^{-1} for the biconjugate gradient method they use to solve it.

The result (B.14) simplifies in the following special cases

- If there is no truncation, i.e. if $k = n$ we have $\Gamma^{\mathrm{tr}}_{\overline{A}} = 0$.
- If the loss function is invariant under the enlarged[1] gauge transformation

$$U \mapsto UQ \qquad , \qquad S \mapsto Q^\dagger S Q \tag{B.17}$$

  for arbitrary unitary Q, we find that $\Gamma^{\mathrm{U}\circ}_{\overline{A}} = 0$. This drastically simplifies the formula *and* removes divergences in F in the presence of degenerate eigenvalues.

B.1.2 Decomposing Differentials

Let us split the differential $\mathrm{d}U$ into components in the space spanned by U and X respectively.

$$\mathrm{d}U \overset{\text{(B.12)}}{=} UU^\dagger \mathrm{d}U + XX^\dagger \mathrm{d}U =: U\mathrm{d}C_1 + \mathrm{d}C_2 \tag{B.18}$$

Now by differentiating the isometry condition (B.10) we find

$$\mathrm{d}C_1^\dagger = -\mathrm{d}C_1 \tag{B.19}$$

and from the orthogonality constraint (B.11) we get

$$\mathrm{d}X^\dagger U = -X^\dagger \mathrm{d}U = -X^\dagger \mathrm{d}C_2. \tag{B.20}$$

Now we define the transformed differential

$$\mathrm{d}P := U^\dagger \mathrm{d}AU \overset{\text{(B.19)}}{=} \mathrm{d}C_1 S + \mathrm{d}S - S\mathrm{d}C_1. \tag{B.21}$$

From this differential we can extract

$$\mathbb{1} \circ \mathrm{d}P \overset{\text{(B.3)}}{=} \mathbb{1} \circ \mathrm{d}S \overset{\text{(B.13)}}{=} \mathrm{d}S, \tag{B.22}$$

as well as with $F_{ij} = 1/(S_i - S_j)$ for $i \neq j$ and $F_{ii} = 0$.

$$F \circ \mathrm{d}P = -\mathcal{O} \circ \mathrm{d}C_1. \tag{B.23}$$

[1] A smaller gauge freedom, where Q is restricted to be diagonal and transforms only the phase of the eigenvectors is inherent to any eigendecomposition. In that sense, the gauge freedom discussed here is “enlarged” to arbitrary unitary Q.

Finally, consider

$$XX^\dagger \mathrm{d}AU \overset{(B.20)}{=} \mathrm{d}C_2 S - XX^\dagger A \mathrm{d}C_2. \tag{B.24}$$

We have found expressions for $\mathrm{d}S$ and the off-diagonal parts of $\mathrm{d}C_1$, as well as an equation that determines $\mathrm{d}C_2$. We are missing an equation for the diagonal parts of $\mathrm{d}C_1$. This is a feature unique to the complex case, since in the real case (B.19) implies that this diagonal part vanishes.

B.1.3 Gauge Invariance

The diagonal parts can be dealt with by exploiting the gauge invariance of any eigendecompostion. The phase of the eigenvalues is arbitrary, such that the transformation $U \mapsto U\Lambda$ for a diagonal unitary Λ preserves the defining properties of the decomposition. As such, any well defined cost function that uses an eigendecomposition as an intermediate step must be invariant under that transformation. Note that we are free to choose the gauge Λ, *and* independently its variations $\mathrm{d}\Lambda$, this is because we can choose a function $\Lambda(A)$ of gauge choices, which has an independent value and derivative at the point of interest.

Since Λ is diagonal $\mathcal{O} \circ \Lambda = 0$ and unitary $\Lambda\Lambda^\dagger = \mathbb{1}$, its variations are constrained by

$$\mathcal{O} \circ \mathrm{d}\Lambda = 0 \qquad ; \qquad \mathrm{d}\Lambda^\dagger \, \Lambda = -\Lambda \mathrm{d}\Lambda^\dagger \tag{B.25}$$

but otherwise arbitrary. On the differential $\mathrm{d}C_1$, the gauge transformation has the following effect

$$\mathrm{d}C_1 = U^\dagger \mathrm{d}U \mapsto \Lambda^\dagger \mathrm{d}C_1 \Lambda + \Lambda^\dagger \mathrm{d}\Lambda. \tag{B.26}$$

It is therefore convenient to choose $\Lambda = \mathbb{1}$, such that this reduces to $\mathrm{d}C_1 \mapsto \mathrm{d}C_1 + \mathrm{d}\Lambda$. We may now choose the gauge variations as $\mathrm{d}\Lambda = -\mathbb{1} \circ \mathrm{d}\tilde{C_1}$, where the tilde denotes the differential in an arbitrary reference gauge. Therefore, in the chosen gauge we have

$$\mathbb{1} \circ \mathrm{d}C_1 = 0. \tag{B.27}$$

B.1.4 Contributions to the adjoint

The starting point for the autodiff formula is equating the following differentials

$$\mathrm{Tr}\left(\Gamma_{\overline{A}} \mathrm{d}A^\dagger + \text{c.c.}\right) = \mathrm{d}\mathcal{L} = \mathrm{Tr}\left(\Gamma_{\overline{U}} \mathrm{d}U^\dagger + \Gamma_{\overline{S}} \mathrm{d}S^\dagger + \text{c.c.}\right). \tag{B.28}$$

First, consider the contribution via $\mathrm{d}S$. We obtain

$$\begin{aligned} \mathrm{Tr}\left(\Gamma_{\overline{S}} \mathrm{d}S^\dagger + \text{c.c.}\right) &\overset{(B.22)}{=} \mathrm{Tr}\left(\Gamma_{\overline{S}}\left(\mathbb{1} \circ \mathrm{d}P^\dagger\right) + \text{c.c.}\right) \overset{(B.7)}{=} \mathrm{Tr}\left(\left(\mathbb{1} \circ \Gamma_{\overline{S}}\right) U^\dagger \mathrm{d}A^\dagger U + \text{c.c.}\right) \\ &= \mathrm{Tr}\left(\Gamma^{\mathrm{S}}_{\overline{A}} \mathrm{d}A^\dagger + \text{c.c.}\right), \quad \text{where} \quad \Gamma^{\mathrm{S}}_{\overline{A}} := U \Gamma_{\overline{S}} U^\dagger \end{aligned} \tag{B.29}$$

and where we used in the last step that $\Gamma_{\overline{S}}$ is diagonal, because S is diagonal.

We consider separately the two contributions via $\mathrm{d}U = U\mathrm{d}C_1 + \mathrm{d}C_2$. For the contribution from $\mathrm{d}C_1$, recall that it is purely off-diagonal because of (B.27) and we know the value of these off-diagonal entries from (B.23). We find

$$\begin{aligned} \mathrm{Tr}\left(\Gamma_{\overline{U}}(U\mathrm{d}C_1)^\dagger + \text{c.c.}\right) &= \mathrm{Tr}\left(U^\dagger\Gamma_{\overline{U}}(F\circ\mathrm{d}P^\dagger) + \text{c.c.}\right) \\ &\overset{\text{(B.7)}}{=} \mathrm{Tr}\left(\left(-F\circ(U^\dagger\Gamma_{\overline{U}})\right)U^\dagger\mathrm{d}A^\dagger U + \text{c.c.}\right) \\ &= \mathrm{Tr}\left(\Gamma_{\overline{A}}^{\mathrm{Uo}}\mathrm{d}A^\dagger + \text{c.c.}\right), \\ &\quad \text{where} \quad \Gamma_{\overline{A}}^{\mathrm{Uo}} := U\left(-F\circ(U^\dagger\Gamma_{\overline{U}})\right)U^\dagger\ . \end{aligned} \tag{B.30}$$

For the final contribution, let γ be a solution to the Sylvester equation $\gamma S - A^\dagger XX^\dagger\gamma = \Gamma_{\overline{U}}$. Then, we find

$$\begin{aligned} \mathrm{Tr}\left(\Gamma_{\overline{U}}\mathrm{d}C_2^\dagger + \text{c.c.}\right) &= \mathrm{Tr}\left(\gamma\left(S\mathrm{d}C_2^\dagger - \mathrm{d}C_2^\dagger A^\dagger XX^\dagger\right) + \text{c.c.}\right) \\ &\overset{\text{(B.24)}}{=} \mathrm{Tr}\left(\gamma U^\dagger\mathrm{d}A^\dagger XX^\dagger + \text{c.c.}\right) \\ &= \mathrm{Tr}\left(\Gamma_{\overline{A}}^{\mathrm{tr}}\mathrm{d}A^\dagger + \text{c.c.}\right), \quad \text{where} \quad \Gamma_{\overline{A}}^{\mathrm{tr}} := XX^\dagger\gamma U^\dagger\ . \end{aligned} \tag{B.31}$$

In summary, we have dealt with all contributions to the RHS of (B.28) and conclude $\Gamma_{\overline{A}} = \Gamma_{\overline{A}}^{\mathrm{S}} + \Gamma_{\overline{A}}^{\mathrm{Uo}} + \Gamma_{\overline{A}}^{\mathrm{tr}}$.

B.1.5 Special cases

The first special case, namely if there is no truncation and $k = n$, implies $X = 0$ and $\Gamma_{\overline{A}}^{\mathrm{tr}} = 0$ follows directly.

Second, consider the case where the cost function is invariant under the enlarged gauge transformation $U \mapsto UQ$ and $S \mapsto Q^\dagger SQ$ with unitary Q. Note that S is now (in general) no longer diagonal. Similar to the arguments of subsection B.1.3, the gauge transformation are unitary $QQ^\dagger = \mathbb{1}_k$ but otherwise arbitrary, such that the variations are constrained by $\mathrm{d}QQ^\dagger + Q\mathrm{d}Q^\dagger = 0$ but otherwise arbitrary. If we now choose $Q = \mathbb{1}$, that is choose the gauge that makes S diagonal, we find that the variations $\mathrm{d}Q = -\mathrm{d}Q^\dagger$ are anti-hermitian. The effect on the differentials is $\mathrm{d}C_1 \mapsto \mathrm{d}C_1 + \mathrm{d}Q$. Since this differential is anti-hermitian by (B.19), we may choose $\mathrm{d}Q$ such that $\mathrm{d}C_1 = 0$. Therefore, we find no contribution from $\mathrm{d}C_1$, that is $\Gamma_{\overline{A}}^{\mathrm{Uo}} = 0$.

There is one caveat; In this gauge, the variations $\mathrm{d}S$ are (in general) no longer diagonal and acquires an off-diagonal part $S\mathrm{d}Q - \mathrm{d}QS$, such that (B.22) and (B.23) do not hold anymore. The result is still valid, however, since (B.21) now directly implies $\mathrm{d}S = \mathrm{d}P$ such that $\mathrm{Tr}\left(\Gamma_{\overline{S}}\mathrm{d}S^\dagger + \text{c.c.}\right) = \mathrm{Tr}\left(U\Gamma_{\overline{S}}U^\dagger\mathrm{d}A^\dagger + \text{c.c.}\right)$ holds anyway and the expression for $\Gamma_{\overline{A}}^{\mathrm{S}}$ remains unchanged.

B.2 Derivations for SVD

The derivations for the SVD are in large parts similar to the derivations for the eigendecomposition above, but also differ substantially. For completeness and readability, we nevertheless give a full derivation again, which has overlap with the previous section.

Let us first reiterate the setup. The truncated SVD of an $m \times n$ matrix A is given by

$$A = USV^\dagger + XYZ^\dagger, \tag{B.32}$$

where S is a $k \times k$ real, positive diagonal matrix – the kept singular values. On the other hand Y is $(m-k) \times (n-k)$ real, non-negative and all entries off the main diagonal vanish – the discarded singular values. Note that we demand that S is strictly positive, i.e. that all vanishing singular values are in Y. This allows for a stable inverse of S, even for rank-deficient A. The matrices U, V, X, Y are (left) isometries

$$\begin{gathered} U^\dagger U = \mathbb{1}_k = V^\dagger V \\ X^\dagger X = \mathbb{1}_{m-k} \quad ; \quad Z^\dagger Z = \mathbb{1}_{n-k} \end{gathered} \tag{B.33}$$

and X (Z) is the orthogonal complement of U (V) such that

$$U^\dagger X = 0 \quad ; \quad V^\dagger Z = 0 \tag{B.34}$$

$$UU^\dagger + XX^\dagger = \mathbb{1}_m \quad ; \quad VV^\dagger + ZZ^\dagger = \mathbb{1}_n \ . \tag{B.35}$$

The result for the AD formula is given in section 4.2.2.

As a standard recipe for deriving AD formulae, we start by equating the expressions for the total derivative of the loss function $\mathcal{L}$, once expressed in terms of the input A and once in terms of the outputs U, S, V.

$$\mathrm{Tr}\left(\Gamma_{\overline{A}} \mathrm{d}A^\dagger + \mathrm{c.c.}\right) = \mathrm{d}\mathcal{L} = \mathrm{Tr}\left(\Gamma_{\overline{U}} \mathrm{d}U^\dagger + \Gamma_{\overline{S}} \mathrm{d}S^\dagger + \Gamma_{\overline{V}} \mathrm{d}V^\dagger + \mathrm{c.c.}\right) \tag{B.36}$$

Our goal is then to solve for an expression of $\Gamma_{\overline{A}}$ in terms of $\Gamma_{\overline{U}}, \Gamma_{\overline{S}}, \Gamma_{\overline{V}}$ and U, S, V, A.

B.2.1 Decomposing Differentials

We start with the differential of (B.32)

$$\mathrm{d}A = \mathrm{d}USV^\dagger + U\mathrm{d}SV^\dagger + US\mathrm{d}V^\dagger + \mathrm{d}XYZ^\dagger + X\mathrm{d}YZ^\dagger + XY\mathrm{d}Z^\dagger. \tag{B.37}$$

We note that since S is real and diagonal, so is $\mathrm{d}S$, i.e.

$$\mathbb{1} \circ \mathrm{d}S = \mathrm{d}S = \mathrm{d}S^\dagger. \tag{B.38}$$

It is convenient to decompose the differentials of the isometries into two parts, using (B.35)

$$\mathrm{d}U = UU^\dagger \mathrm{d}U + XX^\dagger \mathrm{d}U =: U\mathrm{d}C_1 + \mathrm{d}C_2 \tag{B.39}$$

$$\mathrm{d}V = VV^\dagger \mathrm{d}V + ZZ^\dagger \mathrm{d}V =: V\mathrm{d}D_1 + \mathrm{d}D_2 \ . \tag{B.40}$$

By differentiating the isometry constraints (B.33) we find that the first parts are anti-hermitian

$$\mathrm{d}C_1^\dagger = -\mathrm{d}C_1 \quad ; \quad \mathrm{d}D_1^\dagger = -\mathrm{d}D_1. \tag{B.41}$$

The orthogonality constraints (B.34) on the other hand yield

$$\mathrm{d}X^\dagger U = -X^\dagger \mathrm{d}U \quad ; \quad \mathrm{d}Z^\dagger V = -Z^\dagger \mathrm{d}V. \tag{B.42}$$

Finally, we transform/project $\mathrm{d}A$ using (B.33), (B.34) and (B.41)

$$\mathrm{d}P := U^\dagger \mathrm{d}AV = \mathrm{d}C_1 S + \mathrm{d}S - S\mathrm{d}D_1. \tag{B.43}$$

B.2.2 Intermediate Results

Using (B.43), (B.41) and (B.3) we find

$$\mathrm{d}S = \mathbb{1} \circ \frac{\mathrm{d}P + \mathrm{d}P^\dagger}{2} \tag{B.44}$$

$$\mathbb{1} \circ (\mathrm{d}C_1 - \mathrm{d}D_1) = \left[\mathbb{1} \circ \frac{\mathrm{d}P - \mathrm{d}P^\dagger}{2}\right] S^{-1} \; . \tag{B.45}$$

Using (B.43), (B.2) and (B.4) we find

$$\mathcal{O} \circ \left(\mathrm{d}PS + S\mathrm{d}P^\dagger\right) = \mathrm{d}C_1 S^2 - S^2 \mathrm{d}C_1, \tag{B.46}$$

which we can solve for

$$\mathcal{O} \circ \mathrm{d}C_1 = F \circ \left(\mathrm{d}PS + S\mathrm{d}P^\dagger\right). \tag{B.47}$$

Recall that $F_{ij} = 1/(S_i^2 - S_j^2)$ for $i \neq j$ and $F_{ii} = 0$. Similarly, we obtain

$$\mathcal{O} \circ \left(S\mathrm{d}P + \mathrm{d}P^\dagger S\right) = \mathrm{d}D_1 S^2 - S^2 \mathrm{d}D_1 \tag{B.48}$$

$$\mathcal{O} \circ \mathrm{d}D_1 = F \circ \left(S\mathrm{d}P + \mathrm{d}P^\dagger S\right). \tag{B.49}$$

Next use (B.42) to obtain

$$XX^\dagger \mathrm{d}AV = \mathrm{d}C_2 S - XX^\dagger A\mathrm{d}D_2 \tag{B.50}$$

$$ZZ^\dagger \mathrm{d}A^\dagger U = \mathrm{d}D_2 S - ZZ^\dagger A^\dagger \mathrm{d}C_2 \tag{B.51}$$

We have found expressions for $\mathrm{d}S$, as well as for the off-diagonal parts of $\mathrm{d}C_1$ and $\mathrm{d}D_1$, a set of coupled equations that determine $\mathrm{d}C_2$ and $\mathrm{d}D_2$, but only one equation for the diagonal parts of both $\mathrm{d}C_1$ and $\mathrm{d}D_1$. We are missing a second equation to determine these diagonal parts. This is a unique feature of the complex case, since in the real case equation (B.41) implies that the diagonals vanish. The missing equation can be obtained from the gauge invariance inherent to a complex SVD.

B.2.3 Gauge Invariance

The SVD has an inherent gauge freedom of k complex phases, since for a diagonal unitary Λ we have

$$A = USV^\dagger = U\Lambda\Lambda^\dagger S\Lambda\Lambda^\dagger V^\dagger = (U\Lambda)S(V\Lambda)^\dagger, \tag{B.52}$$

such that

$$U \mapsto \tilde{U} := U\Lambda \;,\; S \mapsto S \;,\; V \mapsto \tilde{V} := V\Lambda \tag{B.53}$$

yields another SVD of A that is just as valid. Therefore, a well-defined loss function that uses an SVD as an intermediate step must be invariant under this gauge freedom; $E(U, S, V) = E(U\Lambda, S, V\Lambda)$. Note that we are free to choose the gauge Λ, *and* independently its variations $\mathrm{d}\Lambda$. We can think of a function $\Lambda(M)$ of gauge choices for every possible input M and we may independently choose its value $\Lambda(A)$ at the given input A, and its variations $\mathrm{d}\Lambda = \sum_{ij}(\partial\Lambda/\partial M_{ij})(A)\ \mathrm{d}M_{ij}$.

Since Λ is diagonal $\mathcal{O} \circ \Lambda = 0$ and unitary $\Lambda\Lambda^\dagger = \mathbb{1}$, the variations fulfill

$$\mathcal{O} \circ \mathrm{d}\Lambda = 0 \;,\; \mathrm{d}\Lambda\Lambda^\dagger = -\Lambda\mathrm{d}\Lambda^\dagger. \tag{B.54}$$

On the differential $\mathrm{d}C_1$, the gauge transformation has the following effect

$$\mathrm{d}C_1 = U^\dagger \mathrm{d}U \mapsto \Lambda^\dagger U^\dagger \left(\mathrm{d}U\Lambda + U\mathrm{d}\Lambda\right) = \Lambda^\dagger \mathrm{d}C_1 \Lambda + \Lambda^\dagger \mathrm{d}\Lambda \tag{B.55}$$

It is thus convenient to choose $\Lambda = \mathbb{1}$, such that $\mathrm{d}C_1 \mapsto \mathrm{d}C_1 + \mathrm{d}\Lambda$ simply obtains an additive contribution. In this choice, $\mathrm{d}\Lambda$ is constrained to be diagonal and purely imaginary, but otherwise arbitrary. We can thus choose the gauge variations as $\mathrm{d}\Lambda = -\mathbb{1} \circ \mathrm{d}C_1$ which is diagonal by construction and purely imaginary by (B.41), such that in the new gauge

$$\mathbb{1} \circ \mathrm{d}C_1 = 0. \tag{B.56}$$

B.2.4 Contributions to the Adjoint

We now decompose the differentials in (B.36) as

$$\mathrm{d}U = U\left(\mathbb{1} \circ \mathrm{d}C_1\right) + U\left(\mathcal{O} \circ \mathrm{d}C_1\right) + \mathrm{d}C_2 \tag{B.57}$$

$$\mathrm{d}V = V\left(\mathbb{1} \circ \mathrm{d}D_1\right) + V\left(\mathcal{O} \circ \mathrm{d}D_1\right) + \mathrm{d}D_2, \tag{B.58}$$

which gives us several contributions to $\Gamma_{\overline{A}}$, which we treat separately in the following. First, for the contribution via $\mathrm{d}S$ we use equations (B.44), (B.7), (B.8) to obtain

$$\mathrm{Tr}\left(\Gamma_{\overline{S}} \mathrm{d}S^\dagger + \text{c.c.}\right) = \mathrm{Tr}\left(\Gamma^{\mathrm{S}}_{\overline{A}} \mathrm{d}A^\dagger + \text{c.c.}\right), \tag{B.59}$$

$$\text{where } \Gamma^{\mathrm{S}}_{\overline{A}} := U\left(\mathbb{1} \circ \frac{\Gamma_{\overline{S}} + \Gamma^\dagger_{\overline{S}}}{2}\right) V^\dagger. \tag{B.60}$$

Second, for the off-diagonal contribution via $\mathrm{d}C_1$ use (B.47), (B.7) and (B.8) to obtain

$$\mathrm{Tr}\left(\Gamma_{\overline{U}} \left[U\left(\mathcal{O} \circ \mathrm{d}C_1\right)\right]^\dagger + \text{c.c.}\right) = \mathrm{Tr}\left(\Gamma^{\mathrm{U}\circ}_{\overline{A}} \mathrm{d}A^\dagger + \text{c.c.}\right), \tag{B.61}$$

$$\text{where } \Gamma^{\mathrm{U}\circ}_{\overline{A}} := U\left(J + J^\dagger\right) S V^\dagger \qquad ; \qquad J := F \circ \left(U^\dagger \Gamma_{\overline{U}}\right). \tag{B.62}$$

Third, we get analogously from (B.49)

$$\mathrm{Tr}\left(\Gamma_{\overline{V}} \left[V\left(\mathcal{O} \circ \mathrm{d}D_1\right)\right]^\dagger + \text{c.c.}\right) = \mathrm{Tr}\left(\Gamma^{\mathrm{V}\circ}_{\overline{A}} \mathrm{d}A^\dagger + \text{c.c.}\right), \tag{B.63}$$

$$\text{where } \Gamma^{\mathrm{V}\circ}_{\overline{A}} := US\left(K + K^\dagger\right) V^\dagger \qquad ; \qquad K := F \circ \left(V^\dagger \Gamma_{\overline{V}}\right). \tag{B.64}$$

Fourth, for the diagonal contributions via $\mathrm{d}C_1$ and $\mathrm{d}D_1$ we use (B.45) and (B.56) and obtain

$$\mathrm{Tr}\left(\Gamma_{\overline{U}}\left[U\left(\mathbb{1}\circ\mathrm{d}C_1\right)\right]^\dagger+\Gamma_{\overline{V}}\left[V\left(\mathbb{1}\circ\mathrm{d}D_1\right)\right]^\dagger+\text{c.c.}\right)=\mathrm{Tr}\left(\Gamma_{\overline{A}}^{\text{diag}}\mathrm{d}A^\dagger+\text{c.c.}\right),\tag{B.65}$$

$$\text{where }\Gamma_{\overline{A}}^{\text{diag}}:=\frac{1}{2}US^{-1}\left(L^\dagger-L\right)V^\dagger\qquad;\qquad L:=\mathbb{1}\circ\left(V^\dagger\Gamma_{\overline{V}}\right).\tag{B.66}$$

Finally, the contribution via $\mathrm{d}C_2$ and $\mathrm{d}D_2$, i.e. via the truncated spectrum, is given by

$$\mathrm{Tr}\left(\Gamma_{\overline{U}}\mathrm{d}C_2^\dagger+\Gamma_{\overline{V}}\mathrm{d}D_2^\dagger+\text{c.c.}\right)=\mathrm{Tr}\left(\Gamma_{\overline{A}}^{\text{tr}}\mathrm{d}A^\dagger+\text{c.c.}\right),\tag{B.67}$$

$$\text{with }\Gamma_{\overline{A}}^{\text{tr}}:=XX^\dagger\gamma V^\dagger+U\varphi^\dagger ZZ^\dagger.\tag{B.68}$$

where γ,φ are the solutions of the coupled Sylvester equations

$$\begin{aligned}\Gamma_{\bar{U}}&=\gamma S-AZZ^\dagger\varphi\\ \Gamma_{\bar{V}}&=\varphi S-A^\dagger XX^\dagger\gamma.\end{aligned}\tag{B.69}$$

This can be shown by plugging (B.69) into the LHS of (B.67) and using equations (B.50) and (B.51).

In summary, we have treated all contribution to the RHS of (B.36) such that we have $\Gamma_{\overline{A}}=\Gamma_{\overline{A}}^{\text{S}}+\Gamma_{\overline{A}}^{\text{Uo}}+\Gamma_{\overline{A}}^{\text{Vo}}+\Gamma_{\overline{A}}^{\text{diag}}+\Gamma_{\overline{A}}^{\text{tr}}$.

B.2.5 Special cases

Let us now consider the special cases listed in section 4.2.

Firstly, for a real SVD, the diagonal contribution $\Gamma_{\overline{A}}^{\text{diag}}=0$ vanishes, since (B.41) implies that the diagonals $\mathbb{1}\circ\mathrm{d}C_1=0=\mathbb{1}\circ\mathrm{d}D_1$ vanish.

Secondly, if there is no truncation, that is if $k=\min(m,n)$, we have $X=0$ (if $k=m$) and/or $Z=0$ (if $k=n$). In either case $AZZ^\dagger=0=A^\dagger XX^\dagger$, such that the Sylvester equations (B.69) simplify and admit closed-form solutions $\gamma=\Gamma_{\overline{U}}S^{-1}$ and $\varphi=\Gamma_{\overline{V}}S^{-1}$. We obtain the known AD term for rectangular matrices.

Thirdly, if A is square ($m=n$) and there is no truncation ($m=n=k$), we have that both $X=0=Z$ such that $\Gamma_{\overline{A}}^{\text{tr}}=0$.

Lastly, if the loss function is invariant under the enlarged gauge transformation

$$U\mapsto UQ\qquad;\qquad S\mapsto Q^\dagger SR\qquad;\qquad V\mapsto VR,\tag{B.70}$$

we may proceed similar to the arguments of subsection B.2.3. The gauge transformations Q,V are unitary $QQ^\dagger=\mathbb{1}=RR^\dagger$ but otherwise arbitrary, such that the variations are constrained by $\mathrm{d}QQ^\dagger+Q\mathrm{d}Q^\dagger=0$ but otherwise arbitrary. If we now choose $Q=\mathbb{1}$ we find that $\mathrm{d}Q$ is anti-hermitian $\mathrm{d}Q^\dagger=-\mathrm{d}Q$, and similarly for $\mathrm{d}R$. In the gauge choice $Q=\mathbb{1}=R$, the effect of the gauge variations on the differentials is $\mathrm{d}C_1 \mapsto \mathrm{d}C_1 + \mathrm{d}Q$ and $\mathrm{d}D_1 \mapsto \mathrm{d}D_1 + \mathrm{d}R$. Since those differentials are indeed anti-hermitian by (B.41), we may choose the gauge variations such that in the new gauge $\mathrm{d}C_1 = 0 = \mathrm{d}D_1$. Therefore, we find $\Gamma_{\overline{A}}^{\mathrm{Uo}} = \Gamma_{\overline{A}}^{\mathrm{Vo}} = \Gamma_{\overline{A}}^{\mathrm{diag}} = 0$. Similar to the eigendecomposition case, the differential $\mathrm{d}S$ acquires off-diagonal contribution from the gauge *variations*, even if the gauge is chosen such that S is diagonal. Equation (B.44) no longer holds and only yields the diagonal part of $\mathrm{d}S$. On the other hand, we directly get $\mathrm{d}S = \mathrm{d}P$ from (B.43) such that we find $\mathrm{Tr}\left(\Gamma_{\overline{S}}\mathrm{d}S^\dagger + \mathrm{c.c.}\right) = \mathrm{Tr}\left(U\Gamma_{\overline{S}}V^\dagger\mathrm{d}A^\dagger + \mathrm{c.c.}\right)$ and therefore $\Gamma_{\overline{A}}^{\mathrm{S}} = U\Gamma_{\overline{S}}V^\dagger$ instead.

Acknowledgments

First of all, I would like to thank my supervisor, Frank Pollmann, for your guidance and the countless lessons I learned from you over the years, from specific mechanisms in condensed matter physics, computational methods, and science communication to a healthy perspective on academia as a profession, and for embarking on the next journey with me.

I would also like to thank Johannes Hauschild for many discussions and for showing me the importance of accessibility, good documentation, and open-source culture. I thoroughly enjoyed working with you on TeNPy and hope to continue doing so for a while.

Thank you to Johannes Knolle for agreeing to co-examine my thesis and to Laurens Vanderstraeten and Michael Knap for discussions, feedback, and guidance. I extend my gratitude to all current and past members of the QMQI (formerly CMT) groups at TUM for discussions, sharing your expertise, your kindness, and your jokes on so many occasions. In particular, I thank the PIs for supporting, and Avedis for organizing a reliable supply of caffeine. Thank you Leonie Spitz, Nico Kirchner, Ludwig Zweng, Stefan Birnkammer, ShengHsuan Lin, Norbert Kaiser, Stefan Recksiegel.

I am deeply grateful to everyone who supported me during various stages of my life and my education that have led me to this point, to Wolfgang, to Annika, Maxi, and Philipp, to Ira, and most importantly, to my parents.

Acronyms

AD automatic differentiation. 39, 48, 50, 65, 67–71, 82, 150, 167, 172, 175

ALS alternating least squares. 52

bMPO bulk matrix product operator. 36, 37

bMPS boundary matrix product state. 35–37, 39, 70–72, 75, 77, 78, 82

CPU central processing unit. 4, 48, 55, 63, 81

CTMRG corner transfer renormalization group. 38, 39, 70, 167

DMRG density matrix renormalization group. 4, 5, 7, 8, 12, 19, 23, 24, 26–28, 30, 32, 38, 45, 47, 52, 63, 65, 71, 77, 81, 149, 150

DSF dynamical spin structure factor. 78–80

dSVD deformed singular value decomposition. 15, 18, 21, 22, 31, 37, 49–54, 56, 57, 59, 63, 68–70, 150

ED exact diagonalization. 3

FCS finitely correlated state. 7

FFT fast Fourier transform. 57, 58

FFU fast full update. 39

FLOP floating point operation. 35, 40

FU full update. 7, 38, 39, 77, 81, 82

GPU graphics processing unit. 4, 45, 48, 49, 55, 59, 63, 82, 149

iMPS infinite matrix product state. 7, 32

iPEPS infinite projected entangled pair state. 34, 80, 149, 167

iTEBD infinite time evolving block decimation. 54, 60

L-BFGS Limited memory Broyden–Fletcher–Goldfarb–Shanno. 71

LQ LQ decomposition. 14, 15, 30, 49, 51–54, 56, 58, 142

MERA multiscale entanglement renormalization ansatz. 4, 7, 10, 32

MPO matrix product operator. 19, 23, 24, 26, 32, 34, 36, 45, 56, 73–76, 112

MPO evolution matrix operator based time evolution. 5, 8, 12, 19, 22, 26, 30, 36–38, 45, 63, 66, 149

MPS matrix product state. 3–5, 7, 8, 10–34, 36, 42, 45, 47, 50–52, 54, 60, 62, 63, 66, 77, 81, 144, 145, 148–150

PEPO projected entangled pair operator. 34, 35, 37, 66, 72–77

PEPS projected entangled pair state. 1, 4, 5, 7, 8, 32–39, 65, 66, 68, 70, 71, 77, 78, 80, 81, 112, 149

QLP QLP decomposition. 51, 63, 64, 149

QR QR decomposition. 5, 13–16, 22, 26–28, 30, 31, 37, 47–49, 51–63, 142, 149, 150, 185

QRCP QR decomposition with column pivoting. 50, 51, 64, 149

rSVD randomized singular value decomposition. 5, 51

SRFT subsampled random Fourier transform. 57, 58

SU simple update. 38, 39, 81

SVD singular value decomposition. 4, 5, 11, 15–18, 21, 22, 26, 27, 30, 31, 37, 42, 44, 47–64, 68–70, 82, 142, 147, 149, 150, 167, 171–175

TDVP time dependent variational principle. 12, 19, 22, 45

TEBD time evolving block decimation. 1, 5, 8, 12, 20–24, 45, 47, 51–54, 56, 58, 60–63, 81, 149

TeNPy Tensor Network Python. 1, 5, 6, 8, 45, 54, 56, 77, 84, 142, 144, 147–150

TFIM transverse field Ising model. 60, 73, 75–78, 81

TNS tensor network state. 4, 5, 7, 8, 10, 32–34, 47, 50, 55, 57–59, 63, 77, 80, 92, 139

TPS tensor product state. 4, 10, 65

TPU tensor processing unit. 4, 77, 81

tQR truncated QR-like decomposition. 22, 31, 49, 50, 54, 59

TTN tree tensor network. 4, 7

VUMPS variational uniform matrix product state. 4, 7, 12, 45

List of Algorithms

List of Figures

List of Tables

Bibliography

References [1–4] are given in the list of publications on page 1.

[5] J. G. Bednorz and K. A. Müller, *Possible high Tc superconductivity in the Ba-La-Cu-O system*, en, *Zeitschrift für Physik B Condensed Matter*, vol. 64, no. 2, pp. 189–193, Jun. 1986. DOI: 10.1007/BF01303701.

[6] P. W. Anderson, *The Resonating Valence Bond State in La2CuO4 and Superconductivity*, *Science*, vol. 235, no. 4793, pp. 1196–1198, Mar. 1987. DOI: 10.1126/science.235.4793.1196.

[7] D. C. Tsui, H. L. Stormer, and A. C. Gossard, *Two-Dimensional Magnetotransport in the Extreme Quantum Limit*, *Physical Review Letters*, vol. 48, no. 22, pp. 1559–1562, May 1982. DOI: 10.1103/PhysRevLett.48.1559.

[8] H. L. Stormer, *Nobel Lecture: The fractional quantum Hall effect*, *Reviews of Modern Physics*, vol. 71, no. 4, pp. 875–889, Jul. 1999. DOI: 10.1103/RevModPhys.71.875.

[9] J. Léonard, S. Kim, *et al.*, *Realization of a fractional quantum Hall state with ultracold atoms*, en, *Nature*, vol. 619, no. 7970, pp. 495–499, Jul. 2023. DOI: 10.1038/s41586-023-06122-4.

[10] J. Hubbard, *Electron correlations in narrow energy bands*, *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 276, no. 1365, pp. 238–257, 1963. DOI: 10.1098/rspa.1963.0204.

[11] F. C. Zhang and T. M. Rice, *Effective Hamiltonian for the superconducting Cu oxides*, *Physical Review B*, vol. 37, no. 7, pp. 3759–3761, Mar. 1988. DOI: 10.1103/PhysRevB.37.3759.

[12] E. Dagotto, *Correlated electrons in high-temperature superconductors*, *Reviews of Modern Physics*, vol. 66, no. 3, pp. 763–840, Jul. 1994. DOI: 10.1103/RevModPhys.66.763.

[13] M. Qin, C.-M. Chung, *et al.*, *Absence of Superconductivity in the Pure Two-Dimensional Hubbard Model*, *Physical Review X*, vol. 10, no. 3, p. 031 016, Jul. 2020. DOI: 10.1103/PhysRevX.10.031016.

[14] F. Becca and S. Sorella, *Quantum Monte Carlo approaches for correlated systems*. Cambridge University Press, 2017.

[15] M. Troyer and U.-J. Wiese, *Computational Complexity and Fundamental Limitations to Fermionic Quantum Monte Carlo Simulations*, *Physical Review Letters*, vol. 94, no. 17, p. 170 201, May 2005. DOI: 10.1103/PhysRevLett.94.170201.

[16] M. B. Hastings, *An area law for one-dimensional quantum systems*, en, *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2007, no. 08, P08024, Aug. 2007. DOI: 10.1088/1742-5468/2007/08/P08024.

[17] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac, *Entropy Scaling and Simulability by Matrix Product States*, *Physical Review Letters*, vol. 100, no. 3, p. 030 504, Jan. 2008. DOI: 10.1103/PhysRevLett.100.030504.

[18] D. Gottesman and M. B. Hastings, *Entanglement versus gap for one-dimensional spin systems*, en, *New Journal of Physics*, vol. 12, no. 2, p. 025 002, Feb. 2010. DOI: 10.1088/1367-2630/12/2/025002.

[19] M. Fannes, B. Nachtergaele, and R. F. Werner, *Ground states of VBS models on cayley trees*, en, *Journal of Statistical Physics*, vol. 66, no. 3, pp. 939–973, Feb. 1992. DOI: 10.1007/BF01055710.

[20] Y.-Y. Shi, L.-M. Duan, and G. Vidal, *Classical simulation of quantum many-body systems with a tree tensor network*, *Physical Review A*, vol. 74, no. 2, p. 022 320, Aug. 2006. DOI: 10.1103/PhysRevA.74.022320.

[21] V. Murg, F. Verstraete, Ö. Legeza, and R. M. Noack, *Simulating strongly correlated quantum systems with tree tensor networks*, *Physical Review B*, vol. 82, no. 20, p. 205 105, Nov. 2010. DOI: 10.1103/PhysRevB.82.205105.

[22] G. Vidal, *Entanglement Renormalization*, *Physical Review Letters*, vol. 99, no. 22, p. 220 405, Nov. 2007. DOI: 10.1103/PhysRevLett.99.220405.

[23] G. Vidal, *Class of Quantum Many-Body States That Can Be Efficiently Simulated*, *Physical Review Letters*, vol. 101, no. 11, p. 110 501, Sep. 2008. DOI: 10.1103/PhysRevLett.101.110501.

[24] G. Evenbly and G. Vidal, *Algorithms for entanglement renormalization*, *Physical Review B*, vol. 79, no. 14, p. 144 108, Apr. 2009. DOI: 10.1103/PhysRevB.79.144108.

[25] Y. Nishio, N. Maeshima, A. Gendiar, and T. Nishino, *Tensor Product Variational Formulation for Quantum Systems*, Jan. 2004. DOI: 10.48550/arXiv.cond-mat/0401115.

[26] F. Verstraete and J. I. Cirac, *Renormalization algorithms for Quantum-Many Body Systems in two and higher dimensions*, Jul. 2004. DOI: 10.48550/arXiv.cond-mat/0407066.

[27] J. Dukelsky, M. A. Martín-Delgado, T. Nishino, and G. Sierra, *Equivalence of the variational matrix product method and the density matrix renormalization group applied to spin chains*, en, *Europhysics Letters*, vol. 43, no. 4, p. 457, Aug. 1998. DOI: 10.1209/epl/i1998-00381-x.

[28] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, *Annals of Physics*, January 2011 Special Issue, vol. 326, no. 1, pp. 96–192, Jan. 2011. DOI: 10.1016/j.aop.2010.09.012.

[29] V. Zauner-Stauber, L. Vanderstraeten, *et al.*, *Variational optimization algorithms for uniform matrix product states*, *Physical Review B*, vol. 97, no. 4, p. 045 145, Jan. 2018. DOI: 10.1103/PhysRevB.97.045145.

[30] J. Haegeman, T. J. Osborne, and F. Verstraete, *Post-matrix product state methods: To tangent space and beyond*, *Physical Review B*, vol. 88, no. 7, p. 075 133, Aug. 2013. DOI: 10.1103/PhysRevB.88.075133.

[31] L. Vanderstraeten, J. Haegeman, and F. Verstraete, *Tangent-space methods for uniform matrix product states*, en, *SciPost Physics Lecture Notes*, p. 007, Jan. 2019. DOI: 10.21468/SciPostPhysLectNotes.7.

[32] M. Ganahl, J. Beall, *et al.*, *Density Matrix Renormalization Group with Tensor Processing Units*, *PRX Quantum*, vol. 4, no. 1, p. 010 317, Feb. 2023. DOI: 10.1103/PRXQuantum.4.010317.

[33] A. Menczer and Ö. Legeza, *Massively Parallel Tensor Network State Algorithms on Hybrid CPU-GPU Based Architectures*, May 2023. DOI: 10.48550/arXiv.2305.05581.

[34] A. Menczer, K. Kapás, M. A. Werner, and Ö. Legeza, *Two-dimensional quantum lattice models via mode optimized hybrid CPU-GPU density matrix renormalization group method*, *Physical Review B*, vol. 109, no. 19, p. 195 148, May 2024. DOI: 10.1103/PhysRevB.109.195148.

[35] S. Singh, R. N. C. Pfeifer, and G. Vidal, *Tensor network decompositions in the presence of a global symmetry*, *Physical Review A*, vol. 82, no. 5, p. 050 301, Nov. 2010. DOI: 10.1103/PhysRevA.82.050301.

[36] A. Weichselbaum, *Non-abelian symmetries in tensor networks: A quantum symmetry space approach*, *Annals of Physics*, vol. 327, no. 12, pp. 2972–3047, Dec. 2012. DOI: 10.1016/j.aop.2012.07.009.

[37] H.-J. Liao, J.-G. Liu, L. Wang, and T. Xiang, *Differentiable Programming Tensor Networks*, *Physical Review X*, vol. 9, no. 3, p. 031 041, Sep. 2019. DOI: 10.1103/PhysRevX.9.031041.

[38] J. Hasik, D. Poilblanc, and F. Becca, *Investigation of the Néel phase of the frustrated Heisenberg antiferromagnet by differentiable symmetric tensor networks*, en, *SciPost Physics*, vol. 10, no. 1, p. 012, Jan. 2021. DOI: 10.21468/SciPostPhys.10.1.012.

[39] A. Francuz, N. Schuch, and B. Vanhecke, *Stable and efficient differentiation of tensor network algorithms*, en, Nov. 2023. DOI: 10.48550/arXiv.2311.11894.

[40] J. I. Cirac, D. Pérez-García, N. Schuch, and F. Verstraete, *Matrix product states and projected entangled pair states: Concepts, symmetries, theorems*, *Reviews of Modern Physics*, vol. 93, no. 4, p. 045 003, Dec. 2021. DOI: 10.1103/RevModPhys.93.045003.

[41] H. A. Kramers and G. H. Wannier, *Statistics of the Two-Dimensional Ferromagnet. Part II*, *Physical Review*, vol. 60, no. 3, pp. 263–276, Aug. 1941. DOI: 10.1103/PhysRev.60.263.

[42] R. J. Baxter, *Dimers on a Rectangular Lattice*, *Journal of Mathematical Physics*, vol. 9, no. 4, pp. 650–654, Apr. 1968. DOI: 10.1063/1.1664623.

[43] L. Accardi *et al.*, *Topics in quantum probability*, *Physics Reports*, vol. 77, no. 3, pp. 169–192, 1981.

[44] I. Affleck, T. Kennedy, E. H. Lieb, and H. Tasaki, *Rigorous results on valence-bond ground states in antiferromagnets*, *Physical Review Letters*, vol. 59, no. 7, pp. 799–802, Aug. 1987. DOI: 10.1103/PhysRevLett.59.799.

[45] M. Fannes, B. Nachtergaele, and R. F. Werner, *Exact Antiferromagnetic Ground States of Quantum Spin Chains*, en, *Europhysics Letters*, vol. 10, no. 7, p. 633, Dec. 1989. DOI: 10.1209/0295-5075/10/7/005.

[46] M. Fannes, B. Nachtergaele, and R. F. Werner, *Finitely correlated states on quantum spin chains*, en, *Communications in Mathematical Physics*, vol. 144, no. 3, pp. 443–490, Mar. 1992. DOI: 10.1007/BF02099178.

[47] M. Fannes, B. Nachtergaele, and R. F. Werner, *Finitely Correlated Pure States*, *Journal of Functional Analysis*, vol. 120, no. 2, pp. 511–534, Mar. 1994. DOI: 10.1006/jfan.1994.1041.

[48] S. R. White, *Density matrix formulation for quantum renormalization groups*, *Physical Review Letters*, vol. 69, no. 19, pp. 2863–2866, Nov. 1992. DOI: 10.1103/PhysRevLett.69.2863.

[49] S. R. White, *Density-matrix algorithms for quantum renormalization groups*, *Physical Review B*, vol. 48, no. 14, pp. 10 345–10 356, Oct. 1993. DOI: 10.1103/PhysRevB.48.10345.

[50] S. Östlund and S. Rommer, *Thermodynamic Limit of Density Matrix Renormalization*, *Physical Review Letters*, vol. 75, no. 19, pp. 3537–3540, Nov. 1995. DOI: 10.1103/PhysRevLett.75.3537.

[51] S. Singh, R. N. C. Pfeifer, and G. Vidal, *Tensor network states and algorithms in the presence of a global U(1) symmetry*, *Physical Review B*, vol. 83, no. 11, p. 115 125, Mar. 2011. DOI: 10.1103/PhysRevB.83.115125.

[52] I. P. McCulloch and M. Gulácsi, *The non-Abelian density matrix renormalization group algorithm*, en, *Europhysics Letters*, vol. 57, no. 6, p. 852, Mar. 2002. DOI: 10.1209/epl/i2002-00393-0.

[53] S. Singh and G. Vidal, *Tensor network states and algorithms in the presence of a global SU(2) symmetry*, *Physical Review B*, vol. 86, no. 19, p. 195 114, Nov. 2012. DOI: 10.1103/PhysRevB.86.195114.

[54] J. Motruk, M. P. Zaletel, R. S. K. Mong, and F. Pollmann, *Density matrix renormalization group on a cylinder in mixed real and momentum space*, *Physical Review B*, vol. 93, no. 15, p. 155 139, Apr. 2016. DOI: 10.1103/PhysRevB.93.155139.

[55] G. Ehlers, S. R. White, and R. M. Noack, *Hybrid-space density matrix renormalization group study of the doped two-dimensional Hubbard model*, *Physical Review B*, vol. 95, no. 12, p. 125 125, Mar. 2017. DOI: 10.1103/PhysRevB.95.125125.

[56] S. R. White, *Density matrix renormalization group algorithms with a single center site*, *Physical Review B*, vol. 72, no. 18, p. 180 403, Nov. 2005. DOI: 10.1103/PhysRevB.72.180403.

[57] C. Hubig, I. P. McCulloch, U. Schollwöck, and F. A. Wolf, *Strictly single-site DMRG algorithm with subspace expansion*, *Physical Review B*, vol. 91, no. 15, p. 155 115, Apr. 2015. DOI: 10.1103/PhysRevB.91.155115.

[58] I. P. McCulloch, *Infinite size density matrix renormalization group, revisited*, en, Apr. 2008. DOI: 10.48550/arXiv.0804.2509.

[59] L. Vanderstraeten, M. Mariën, F. Verstraete, and J. Haegeman, *Excitations and the tangent space of projected entangled-pair states*, *Physical Review B*, vol. 92, no. 20, p. 201 111, Nov. 2015. DOI: 10.1103/PhysRevB.92.201111.

[60] F. Verstraete, J. J. García-Ripoll, and J. I. Cirac, *Matrix Product Density Operators: Simulation of Finite-Temperature and Dissipative Systems*, *Phys-*

ical Review Letters, vol. 93, no. 20, p. 207 204, Nov. 2004. DOI: 10.1103/PhysRevLett.93.207204.

[61] T. Barthel, U. Schollwöck, and S. R. White, *Spectral functions in one-dimensional quantum systems at finite temperature using the density matrix renormalization group*, *Physical Review B*, vol. 79, no. 24, p. 245 101, Jun. 2009. DOI: 10.1103/PhysRevB.79.245101.

[62] S. Paeckel, T. Köhler, *et al.*, *Time-evolution methods for matrix-product states*, *Annals of Physics*, vol. 411, p. 167 998, Dec. 2019. DOI: 10.1016/j.aop.2019.167998.

[63] A. Dang, C. D. Hill, and L. C. L. Hollenberg, *Optimising Matrix Product State Simulations of Shor's Algorithm*, en-GB, *Quantum*, vol. 3, p. 116, Jan. 2019. DOI: 10.22331/q-2019-01-25-116.

[64] J. Tindall, M. Fishman, E. M. Stoudenmire, and D. Sels, *Efficient Tensor Network Simulation of IBM's Eagle Kicked Ising Experiment*, *PRX Quantum*, vol. 5, no. 1, p. 010 308, Jan. 2024. DOI: 10.1103/PRXQuantum.5.010308.

[65] M. C. Bañuls, R. Orús, *et al.*, *Simulation of many-qubit quantum computation with matrix product states*, *Physical Review A*, vol. 73, no. 2, p. 022 344, Feb. 2006. DOI: 10.1103/PhysRevA.73.022344.

[66] T. Nguyen, D. Lyakh, *et al.*, *Tensor Network Quantum Virtual Machine for Simulating Quantum Circuits at Exascale*, *ACM Transactions on Quantum Computing*, vol. 4, no. 1, 6:1–6:21, Oct. 2022. DOI: 10.1145/3547334.

[67] L. Tagliacozzo, G. Evenbly, and G. Vidal, *Simulation of two-dimensional quantum systems using a tree tensor network that exploits the entropic area law*, *Physical Review B*, vol. 80, no. 23, p. 235 127, Dec. 2009. DOI: 10.1103/PhysRevB.80.235127.

[68] L. Cincio, J. Dziarmaga, and M. M. Rams, *Multiscale Entanglement Renormalization Ansatz in Two Dimensions: Quantum Ising Model*, *Physical Review Letters*, vol. 100, no. 24, p. 240 603, Jun. 2008. DOI: 10.1103/PhysRevLett.100.240603.

[69] S. Yang, Z.-C. Gu, and X.-G. Wen, *Loop Optimization for Tensor Network Renormalization*, *Physical Review Letters*, vol. 118, no. 11, p. 110 504, Mar. 2017. DOI: 10.1103/PhysRevLett.118.110504.

[70] M. P. Zaletel and F. Pollmann, *Isometric Tensor Network States in Two Dimensions*, *Physical Review Letters*, vol. 124, no. 3, p. 037 201, Jan. 2020. DOI: 10.1103/PhysRevLett.124.037201.

[71] S.-H. Lin, M. Zaletel, and F. Pollmann, *Efficient Simulation of Dynamics in Two-Dimensional Quantum Spin Systems with Isometric Tensor Networks*, Nov. 2022. DOI: 10.1103/PhysRevB.106.245102.

[72] G. Evenbly, *Gauge fixing, canonical forms, and optimal truncations in tensor networks with closed loops*, *Physical Review B*, vol. 98, no. 8, p. 085 155, Aug. 2018. DOI: 10.1103/PhysRevB.98.085155.

[73] R. Orús, *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*, *Annals of Physics*, vol. 349, pp. 117–158, Oct. 2014. DOI: 10.1016/j.aop.2014.06.013.

[74] M. C. Bañuls, *Tensor Network Algorithms: A Route Map*, en, *Annual Review of Condensed Matter Physics*, vol. 14, no. Volume 14, 2023, pp. 173–191, Mar. 2023. DOI: 10.1146/annurev-conmatphys-040721-022705.

[75] C. H. Bennett, D. P. DiVincenzo, *et al.*, *Remote State Preparation*, *Physical Review Letters*, vol. 87, no. 7, p. 077 902, Jul. 2001. DOI: 10.1103/PhysRevLett.87.077902.

[76] C. H. Bennett and G. Brassard, *Quantum cryptography: Public key distribution and coin tossing*, *Theoretical Computer Science*, vol. 560, pp. 7–11, 2014. DOI: https://doi.org/10.1016/j.tcs.2014.05.025.

[77] E. Chitambar and G. Gour, *Quantum resource theories*, *Reviews of Modern Physics*, vol. 91, no. 2, p. 025 001, Apr. 2019. DOI: 10.1103/RevModPhys.91.025001.

[78] P. Calabrese and A. Lefevre, *Entanglement spectrum in one-dimensional systems*, *Physical Review A*, vol. 78, no. 3, p. 032 329, Sep. 2008. DOI: 10.1103/PhysRevA.78.032329.

[79] L. Tagliacozzo, T. R. de Oliveira, S. Iblisdir, and J. I. Latorre, *Scaling of entanglement support for matrix product states*, *Physical Review B*, vol. 78, no. 2, p. 024 410, Jul. 2008. DOI: 10.1103/PhysRevB.78.024410.

[80] F. Pollmann, S. Mukerjee, A. M. Turner, and J. E. Moore, *Theory of Finite-Entanglement Scaling at One-Dimensional Quantum Critical Points*, *Physical Review Letters*, vol. 102, no. 25, p. 255 701, Jun. 2009. DOI: 10.1103/PhysRevLett.102.255701.

[81] M. Levin and X.-G. Wen, *Detecting Topological Order in a Ground State Wave Function*, *Physical Review Letters*, vol. 96, no. 11, p. 110 405, Mar. 2006. DOI: 10.1103/PhysRevLett.96.110405.

[82] A. Kitaev and J. Preskill, *Topological Entanglement Entropy*, *Physical Review Letters*, vol. 96, no. 11, p. 110 404, Mar. 2006. DOI: 10.1103/PhysRevLett.96.110404.

[83] H. Li and F. D. M. Haldane, *Entanglement Spectrum as a Generalization of Entanglement Entropy: Identification of Topological Order in Non-Abelian Fractional Quantum Hall Effect States*, *Physical Review Letters*, vol. 101, no. 1, p. 010 504, Jul. 2008. DOI: 10.1103/PhysRevLett.101.010504.

[84] D. N. Page, *Average entropy of a subsystem*, *Physical Review Letters*, vol. 71, no. 9, pp. 1291–1294, Aug. 1993. DOI: 10.1103/PhysRevLett.71.1291.

[85] J. Haegeman, J. I. Cirac, *et al.*, *Time-Dependent Variational Principle for Quantum Lattices*, *Physical Review Letters*, vol. 107, no. 7, p. 070 601, Aug. 2011. DOI: 10.1103/PhysRevLett.107.070601.

[86] J. Haegeman, C. Lubich, *et al.*, *Unifying time evolution and optimization with matrix product states*, *Physical Review B*, vol. 94, no. 16, p. 165 116, Oct. 2016. DOI: 10.1103/PhysRevB.94.165116.

[87] R. Orús and G. Vidal, *Infinite time-evolving block decimation algorithm beyond unitary evolution*, *Physical Review B*, vol. 78, no. 15, p. 155 117, Oct. 2008. DOI: 10.1103/PhysRevB.78.155117.

[88] F. Verstraete and J. I. Cirac, *Matrix product states represent ground states faithfully*, *Physical Review B*, vol. 73, no. 9, p. 094 423, Mar. 2006. DOI: 10.1103/PhysRevB.73.094423.

[89] G. M. Crosswhite and D. Bacon, *Finite automata for caching in matrix product algorithms*, *Physical Review A*, vol. 78, no. 1, p. 012 356, Jul. 2008. DOI: 10.1103/PhysRevA.78.012356.

[90] G. M. Crosswhite, A. C. Doherty, and G. Vidal, *Applying matrix product operators to model systems with long-range interactions*, *Physical Review B*, vol. 78, no. 3, p. 035 116, Jul. 2008. DOI: 10.1103/PhysRevB.78.035116.

[91] B. Pirvu, V. Murg, J. I. Cirac, and F. Verstraete, *Matrix product operator representations*, en, *New Journal of Physics*, vol. 12, no. 2, p. 025 012, Feb. 2010. DOI: 10.1088/1367-2630/12/2/025012.

[92] S. Paeckel, T. Köhler, and S. R. Manmana, *Automated construction of $U(1)$-invariant matrix-product operators from graph representations*, en, *SciPost Physics*, vol. 3, no. 5, p. 035, Nov. 2017. DOI: 10.21468/SciPostPhys.3.5.035.

[93] M. P. Zaletel, R. S. K. Mong, *et al.*, *Time-evolving a matrix product state with long-ranged interactions*, *Physical Review B*, vol. 91, no. 16, p. 165 112, Apr. 2015. DOI: 10.1103/PhysRevB.91.165112.

[94] G. Vidal, *Classical Simulation of Infinite-Size Quantum Lattice Systems in One Spatial Dimension*, *Physical Review Letters*, vol. 98, no. 7, p. 070 201, Feb. 2007. DOI: 10.1103/PhysRevLett.98.070201.

[95] S. Liang and H. Pang, *Approximate diagonalization using the density matrix renormalization-group method: A two-dimensional-systems perspective*, *Physical Review B*, vol. 49, no. 13, pp. 9214–9217, Apr. 1994. DOI: 10.1103/PhysRevB.49.9214.

[96] E. M. Stoudenmire and S. R. White, *Studying Two-Dimensional Systems with the Density Matrix Renormalization Group*, en, *Annual Review of Condensed Matter Physics*, vol. 3, no. Volume 3, 2012, pp. 111–128, Mar. 2012. DOI: 10.1146/annurev-conmatphys-020911-125018.

[97] B. Pirvu, G. Vidal, F. Verstraete, and L. Tagliacozzo, *Matrix product states for critical spin chains: Finite-size versus finite-entanglement scaling*, *Physical Review B*, vol. 86, no. 7, p. 075 117, Aug. 2012. DOI: 10.1103/PhysRevB.86.075117.

[98] A. J. Daley, C. Kollath, U. Schollwöck, and G. Vidal, *Time-dependent density-matrix renormalization-group using adaptive effective Hilbert spaces*, en, *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2004, no. 04, P04005, Apr. 2004. DOI: 10.1088/1742-5468/2004/04/P04005.

[99] S. R. White and A. E. Feiguin, *Real-Time Evolution Using the Density Matrix Renormalization Group*, *Physical Review Letters*, vol. 93, no. 7, p. 076 401, Aug. 2004. DOI: 10.1103/PhysRevLett.93.076401.

[100] G. Vidal, *Efficient Simulation of One-Dimensional Quantum Many-Body Systems*, *Physical Review Letters*, vol. 93, no. 4, p. 040 502, Jul. 2004. DOI: 10.1103/PhysRevLett.93.040502.

[101] A. G. Grushin, J. Motruk, M. P. Zaletel, and F. Pollmann, *Characterization and stability of a fermionic $\ensuremath{\nu}=1/3$ fractional Chern insulator*, *Physical Review B*, vol. 91, no. 3, p. 035 136, Jan. 2015. DOI: 10.1103/PhysRevB.91.035136.

[102] M. P. Zaletel, R. S. K. Mong, and F. Pollmann, *Flux insertion, entanglement, and quantized responses*, en, *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2014, no. 10, P10007, Oct. 2014. DOI: 10.1088/1742-5468/2014/10/P10007.

[103] S.-S. Gong, W. Zhu, and D. N. Sheng, *Emergent Chiral Spin Liquid: Fractional Quantum Hall Effect in a Kagome Heisenberg Model*, en, *Scientific Reports*, vol. 4, no. 1, p. 6317, Sep. 2014. DOI: 10.1038/srep06317.

[104] P. Corboz, S. R. White, G. Vidal, and M. Troyer, *Stripes in the two-dimensional t-J model with infinite projected entangled-pair states*, *Physical Review B*, vol. 84, no. 4, p. 041 108, Jul. 2011. DOI: 10.1103/PhysRevB.84.041108.

[105] P. Corboz, *Improved energy extrapolation with infinite projected entangled-pair states applied to the two-dimensional Hubbard model*, *Physical Review B*, vol. 93, no. 4, p. 045 116, Jan. 2016. DOI: 10.1103/PhysRevB.93.045116.

[106] B.-X. Zheng, C.-M. Chung, *et al.*, *Stripe order in the underdoped region of the two-dimensional Hubbard model*, *Science*, vol. 358, no. 6367, pp. 1155–1160, Dec. 2017. DOI: 10.1126/science.aam7127.

[107] M. Lubasch, J. I. Cirac, and M.-C. Bañuls, *Algorithms for finite projected entangled pair states*, *Physical Review B*, vol. 90, no. 6, p. 064 425, Aug. 2014. DOI: 10.1103/PhysRevB.90.064425.

[108] H. N. Phien, J. A. Bengua, *et al.*, *Infinite projected entangled pair states algorithm improved: Fast full update and gauge fixing*, *Physical Review B*, vol. 92, no. 3, p. 035 142, Jul. 2015. DOI: 10.1103/PhysRevB.92.035142.

[109] P. C. G. Vlaar and P. Corboz, *Simulation of three-dimensional quantum systems with projected entangled-pair states*, *Physical Review B*, vol. 103, no. 20, p. 205 137, May 2021. DOI: 10.1103/PhysRevB.103.205137.

[110] P. C. G. Vlaar and P. Corboz, *Efficient Tensor Network Algorithm for Layered Systems*, *Physical Review Letters*, vol. 130, no. 13, p. 130 601, Mar. 2023. DOI: 10.1103/PhysRevLett.130.130601.

[111] S. S. Jahromi and R. Orús, *Universal tensor-network algorithm for any infinite lattice*, *Physical Review B*, vol. 99, no. 19, p. 195 105, May 2019. DOI: 10.1103/PhysRevB.99.195105.

[112] J. Jordan, R. Orús, *et al.*, *Classical Simulation of Infinite-Size Quantum Lattice Systems in Two Spatial Dimensions*, *Physical Review Letters*, vol. 101, no. 25, p. 250 602, Dec. 2008. DOI: 10.1103/PhysRevLett.101.250602.

[113] D. Perez-Garcia, M. Sanz, *et al.*, *A canonical form for Projected Entangled Pair States and applications*, *New Journal of Physics*, vol. 12, no. 2, p. 025 010, Feb. 2010. DOI: 10.1088/1367-2630/12/2/025010.

[114] A. Acuaviva, V. Makam, *et al.*, *The minimal canonical form of a tensor network*, in *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, Nov. 2023, pp. 328–362. DOI: 10.1109/FOCS57990.2023.00027.

[115] M. Scheb and R. M. Noack, *Finite projected entangled pair states for the Hubbard model*, *Physical Review B*, vol. 107, no. 16, p. 165 112, Apr. 2023. DOI: 10.1103/PhysRevB.107.165112.

[116] N. Chepiga and S. R. White, *Comb tensor networks*, en, *Physical Review B*, vol. 99, no. 23, p. 235 426, Jun. 2019. DOI: 10.1103/PhysRevB.99.235426.

[117] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac, *Computational Complexity of Projected Entangled Pair States*, *Physical Review Letters*, vol. 98, no. 14, p. 140 506, Apr. 2007. DOI: 10.1103/PhysRevLett.98.140506.

[118] L. Wang, I. Pižorn, and F. Verstraete, *Monte Carlo simulation with tensor network states*, *Physical Review B*, vol. 83, no. 13, p. 134 421, Apr. 2011. DOI: 10.1103/PhysRevB.83.134421.

[119] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac, *Simulation of Quantum Many-Body Systems with Strings of Operators and Monte Carlo Tensor Contractions*, *Physical Review Letters*, vol. 100, no. 4, p. 040 501, Jan. 2008. DOI: 10.1103/PhysRevLett.100.040501.

[120] M. Lubasch, J. I. Cirac, and M.-C. Bañuls, *Unifying projected entangled pair state contractions*, en, *New Journal of Physics*, vol. 16, no. 3, p. 033 014, Mar. 2014. DOI: 10.1088/1367-2630/16/3/033014.

[121] J. I. Cirac, D. Poilblanc, N. Schuch, and F. Verstraete, *Entanglement spectrum and boundary theories with projected entangled-pair states*, *Physical Review B*, vol. 83, no. 24, p. 245 134, Jun. 2011. DOI: 10.1103/PhysRevB.83.245134.

[122] T. Nishino and K. Okunishi, *Corner Transfer Matrix Renormalization Group Method*, *Journal of the Physical Society of Japan*, vol. 65, no. 4, pp. 891–894, Apr. 1996. DOI: 10.1143/JPSJ.65.891.

[123] P. Corboz, R. Orús, B. Bauer, and G. Vidal, *Simulation of strongly correlated fermions in two spatial dimensions with fermionic projected entangled-pair states*, *Physical Review B*, vol. 81, no. 16, p. 165 104, Apr. 2010. DOI: 10.1103/PhysRevB.81.165104.

[124] H. C. Jiang, Z. Y. Weng, and T. Xiang, *Accurate Determination of Tensor Network State of Quantum Lattice Models in Two Dimensions*, *Physical Review Letters*, vol. 101, no. 9, p. 090 603, Aug. 2008. DOI: 10.1103/PhysRevLett.101.090603.

[125] W.-Y. Liu, S.-J. Dong, *et al.*, *Gradient optimization of finite projected entangled pair states*, *Physical Review B*, vol. 95, no. 19, p. 195 154, May 2017. DOI: 10.1103/PhysRevB.95.195154.

[126] W.-Y. Liu, Y.-Z. Huang, S.-S. Gong, and Z.-C. Gu, *Accurate simulation for finite projected entangled pair states in two dimensions*, *Physical Review B*, vol. 103, no. 23, p. 235 155, Jun. 2021. DOI: 10.1103/PhysRevB.103.235155.

[127] T. Vieijra, J. Haegeman, F. Verstraete, and L. Vanderstraeten, *Direct sampling of projected entangled-pair states*, *Physical Review B*, vol. 104, no. 23, p. 235 141, Dec. 2021. DOI: 10.1103/PhysRevB.104.235141.

[128] M. J. O'Rourke and G. K.-L. Chan, *Entanglement in the quantum phases of an unfrustrated Rydberg atom array*, *Nature Communications*, vol. 14, no. 1, p. 5397, Sep. 2023. DOI: 10.1038/s41467-023-41166-0.

[129] L. Vanderstraeten, J. Haegeman, P. Corboz, and F. Verstraete, *Gradient methods for variational optimization of projected entangled-pair states*, *Physical Review B*, vol. 94, no. 15, p. 155 123, Oct. 2016. DOI: 10.1103/PhysRevB.94.155123.

[130] P. Czarnik, J. Dziarmaga, and P. Corboz, *Time evolution of an infinite projected entangled pair state: An efficient algorithm*, *Physical Review B*, vol. 99, no. 3, p. 035 115, Jan. 2019. DOI: 10.1103/PhysRevB.99.035115.

[131] J. Dziarmaga, *Time evolution of an infinite projected entangled pair state: Neighborhood tensor update*, *Physical Review B*, vol. 104, no. 9, p. 094 411, Sep. 2021. DOI: 10.1103/PhysRevB.104.094411.

[132] R. Orús and G. Vidal, *Simulation of two-dimensional quantum systems on an infinite lattice revisited: Corner transfer matrix for tensor contraction*, *Physical Review B*, vol. 80, no. 9, p. 094 403, Sep. 2009. DOI: 10.1103/PhysRevB.80.094403.

[133] J. D. A. Espinoza and P. Corboz, *Spectral functions with infinite projected entangled-pair states*, May 2024. DOI: 10.48550/arXiv.2405.10628.

[134] C. Hubig, A. Bohrdt, *et al.*, *Evaluation of time-dependent correlators after a local quench in iPEPS: Hole motion in the t-J model*, en, *SciPost Physics*, vol. 8, no. 2, p. 021, Feb. 2020. DOI: 10.21468/SciPostPhys.8.2.021.

[135] J. Hauschild and F. Pollmann, *Efficient numerical simulations with Tensor Networks: Tensor Network Python (TeNPy)*, en, *SciPost Physics Lecture Notes*, p. 005, Oct. 2018. DOI: 10.21468/SciPostPhysLectNotes.5.

[136] J. A. Kjäll, M. P. Zaletel, *et al.*, *The phase diagram of the anisotropic spin-2 XXZ model: An infinite system DMRG study*, *Physical Review B*, vol. 87, no. 23, p. 235 106, Jun. 2013. DOI: 10.1103/PhysRevB.87.235106.

[137] M. P. Zaletel, R. S. K. Mong, and F. Pollmann, *Topological characterization of fractional quantum Hall ground states from microscopic Hamiltonians*, *Physical Review Letters*, vol. 110, no. 23, p. 236 801, Jun. 2013. DOI: 10.1103/PhysRevLett.110.236801.

[138] P. Businger and G. H. Golub, *Linear least squares solutions by householder transformations*, en, *Numerische Mathematik*, vol. 7, no. 3, pp. 269–276, Jun. 1965. DOI: 10.1007/BF01436084.

[139] T. F. Chan, *Rank revealing QR factorizations*, *Linear Algebra and its Applications*, vol. 88-89, pp. 67–82, Apr. 1987. DOI: 10.1016/0024-3795(87)90103-0.

[140] G. H. Golub and C. F. Van Loan, *Matrix computations*, Fourth edition. JHU press, 2013.

[141] J. A. Duersch and M. Gu, *Randomized QR with Column Pivoting*, *SIAM Journal on Scientific Computing*, vol. 39, no. 4, pp. C263–C291, Jan. 2017. DOI: 10.1137/15M1044680.

[142] J. Xiao, M. Gu, and J. Langou, *Fast Parallel Randomized QR with Column Pivoting Algorithms for Reliable Low-rank Matrix Approximations*, in *2017 IEEE 24th International Conference on High Performance Computing (HiPC)*, Dec. 2017, pp. 233–242. DOI: 10.1109/HiPC.2017.00035.

[143] M. Melnichenko, O. Balabanov, *et al.*, *CholeskyQR with Randomization and Pivoting for Tall Matrices (CQRRPT)*, Feb. 2024. DOI: 10.48550/arXiv.2311.08316.

[144] G. W. Stewart, *The QLP Approximation to the Singular Value Decomposition*, *SIAM Journal on Scientific Computing*, vol. 20, no. 4, pp. 1336–1348, Jan. 1999. DOI: 10.1137/S1064827597319519.

[145] S. Voronin and P.-G. Martinsson, *RSVDPACK: An implementation of randomized algorithms for computing the singular value, interpolative, and CUR decompositions of matrices on multi-core and GPU architectures*, en, Aug. 2016.

[146] N.-C. Wu and H. Xiang, *Randomized QLP decomposition*, *Linear Algebra and its Applications*, vol. 599, Apr. 2020. DOI: 10.1016/j.laa.2020.03.041.

[147] M. F. Kaloorazi, K. Liu, *et al.*, *Randomized Rank-Revealing QLP for Low-Rank Matrix Decomposition*, *IEEE Access*, vol. 11, pp. 63 650–63 666, 2023. DOI: 10.1109/ACCESS.2023.3288889.

[148] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions*, *SIAM Review*, vol. 53, no. 2, pp. 217–288, Jan. 2011. DOI: 10.1137/090771806.

[149] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, *A fast randomized algorithm for the approximation of matrices*, *Applied and Computational Harmonic Analysis*, vol. 25, no. 3, pp. 335–366, Nov. 2008. DOI: 10.1016/j.acha.2007.12.002.

[150] G. Ortiz, E. Cobanera, and Z. Nussinov, *Dualities and the phase diagram of the p-clock model*, *Nuclear Physics B*, vol. 854, no. 3, pp. 780–814, Jan. 2012. DOI: 10.1016/j.nuclphysb.2011.09.012.

[151] G. Sun, T. Vekua, E. Cobanera, and G. Ortiz, *Phase transitions in the ${\mathbb{Z}}_{p}$ and U(1) clock models*, *Physical Review B*, vol. 100, no. 9, p. 094 428, Sep. 2019. DOI: 10.1103/PhysRevB.100.094428.

[152] M. Hefel, *Fast Matrix Product Operator Based Time Evolution*, 2023.

[153] I. P. McCulloch and J. J. Osborne, *Comment on "Controlled Bond Expansion for Density Matrix Renormalization Group Ground State Search at Single-Site Costs" (Extended Version)*, en, Apr. 2024.

[154] J. Hasik, *Towards next-generation methods to optimize two-dimensional tensor networks: Algorithmic differentiation and applications to quantum magnets*, 2019.

[155] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, *Automatic differentiation in machine learning: A survey*, Feb. 2018. DOI: 10.48550/arXiv.1502.05767.

[156] C. C. Margossian, *A review of automatic differentiation and its efficient implementation*, en, *WIREs Data Mining and Knowledge Discovery*, vol. 9, no. 4, e1305, 2019. DOI: 10.1002/widm.1305.

[157] Z.-Q. Wan and S.-X. Zhang, *Automatic Differentiation for Complex Valued SVD*, Nov. 2019. DOI: 10.48550/arXiv.1909.02659.

[158] M. Strathern, *'Improving ratings': Audit in the British University system*, *European review*, vol. 5, no. 3, pp. 305–321, 1997.

[159] H. W. J. Blöte and Y. Deng, *Cluster Monte Carlo simulation of the transverse Ising model*, *Physical Review E*, vol. 66, no. 6, p. 066 110, Dec. 2002. DOI: 10.1103/PhysRevE.66.066110.

[160] L. Vanderstraeten, J. Haegeman, and F. Verstraete, *Simulating excitation spectra with projected entangled-pair states*, *Physical Review B*, vol. 99, no. 16, p. 165 121, Apr. 2019. DOI: 10.1103/PhysRevB.99.165121.

[161] B. Ponsioen and P. Corboz, *Excitations with projected entangled pair states using the corner transfer matrix method*, *Physical Review B*, vol. 101, no. 19, p. 195 109, May 2020. DOI: 10.1103/PhysRevB.101.195109.

[162] P. Schmoll, S. Singh, M. Rizzi, and R. Orus, *A programming guide for tensor networks with global $SU(2)$ symmetry*, en, Sep. 2018. DOI: 10.1016/j.aop.2020.168232.

[163] W. Ji and X.-G. Wen, *Categorical symmetry and noninvertible anomaly in symmetry-breaking and topological phase transitions*, en, *Physical Review Research*, vol. 2, no. 3, p. 033 417, Sep. 2020. DOI: 10.1103/PhysRevResearch.2.033417.

[164] S. H. Simon and S. H. Simon, *Topological Quantum*. Oxford, New York: Oxford University Press, Sep. 2023.

[165] *TensorKit.jl Documentation*DOI: : https://jutho.github.io/TensorKit.jl/stable/.

[166] C. Heunen and J. Vicary, *Categories for Quantum Theory: An Introduction*, en. Oxford University Press, Nov. 2019.

[167] P. Etingof, S. Gelaki, D. Nikshych, and V. Ostrik, *Tensor categories* (Mathematical surveys and monographs volume 205), en. Providence, Rhode Island: American Mathematical Society, 2015.

[168] P. Selinger, *A Survey of Graphical Languages for Monoidal Categories*, en, in *New Structures for Physics*, B. Coecke, Ed., Berlin, Heidelberg: Springer, 2011, pp. 289–355. DOI: 10.1007/978-3-642-12821-9_4.

[169] N. Bultinck, M. Mariën, *et al.*, *Anyons and matrix product operator algebras*, *Annals of Physics*, vol. 378, pp. 183–233, Mar. 2017. DOI: 10.1016/j.aop.2017.01.004.

[170] L. Kong and Z.-H. Zhang, *An invitation to topological orders and category theory*, en, May 2022. DOI: 10.48550/arXiv.2205.05565.

[171] S. Mac Lane, *Natural associativity and commutativity*, *Rice University Studies*, vol. 49, no. 4, pp. 28–46, 1963.

[172] S. Mac Lane, *Categories for the working mathematician*. Springer Science & Business Media, 2013, vol. 5.

[173] A. Feiguin, S. Trebst, *et al.*, *Interacting Anyons in Topological Quantum Liquids: The Golden Chain*, en, *Physical Review Letters*, vol. 98, no. 16, p. 160 409, Apr. 2007. DOI: 10.1103/PhysRevLett.98.160409.

[174] L. Lootens, C. Delcamp, G. Ortiz, and F. Verstraete, *Dualities in One-Dimensional Quantum Lattice Models: Symmetric Hamiltonians and Matrix Product Operator Intertwiners*, en, *PRX Quantum*, vol. 4, no. 2, p. 020 357, Jun. 2023. DOI: 10.1103/PRXQuantum.4.020357.

[175] J. Rasch and A. C. H. Yu, *Efficient Storage Scheme for Precalculated Wigner 3j, 6j and Gaunt Coefficients*, *SIAM Journal on Scientific Computing*, vol. 25, no. 4, pp. 1416–1428, Jan. 2004. DOI: 10.1137/S1064827503422932.