Technische Universität München
Fakultät für Physik

Abschlussarbeit im Masterstudiengang Physik der Kondensierten Materie

Entwicklung eines diagonalen isometrischen Tensor Netzwerk Algorithmus

Development of a diagonal isometric Tensor Network Algorithm

Benjamin Sappler

18. April 2024

Erstgutachter (Themensteller): Prof. F. Pollmann
Zweitgutachter: Unknown

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Munich, 99.99.2099 Benjamin Sappler

Abstract

The numerical simulation of strongly interacting quantum many-body systems is a challenging problem. In the last decades, Tensor Networks have emerged as the standard method for tackling this problem in one dimensional systems in the form of Matrix Product States (MPS). Tensor Networks have also been generalized for the highly relevant problem of two and more spatial dimensions. However, these so-called Projected Entangled Pair States (PEPS) are typically plagued by high computational complexity or drastic approximations. Recently, a new class of Tensor Networks, called isometric Tensor Networks, have been proposed for the simulation of two-dimensional quantum systems. This new class of Tensor Networks can be understood as a generalization of the one-dimensional Matrix Product States to higher dimensions. While isometric Tensor Networks generally capture only a subspace of the total Hilbert space, there are already promising results. In this work, we develop a new class of isometric Tensor Networks that has some key differences to the existing one. We show first numerical results for finding ground states of the Transverse Field Ising model.

Zusammenfassung

Contents

Chapter 1

Introduction

Chapter 2

Tensors and Tensor Networks

2.1 Conventions and Notation

For the purpose of this thesis a *tensor* $\boldsymbol{T}$ *of rank* n is an n-dimensional array of complex numbers

$$\boldsymbol{T} \in \mathbb{C}^{\chi_1 \times \chi_2 \times \dots \times \chi_n}, \quad \chi_i \in \{1, 2, \dots\}$$

with entries

$$T_{i_1 i_2 \dots i_n} \in \mathbb{C}, \quad i_j \in \{1, 2, \dots, \chi_j\}.$$

With this definition, a rank-0 tensor is a scalar, a rank-1 tensor is a vector, and a rank-2 tensor is a matrix. It is convenient to introduce a diagrammatic notation, where tensors are drawn as shapes and tensor indices are drawn as lines (*legs*, *bonds*) emerging from the shapes. To relate this diagrammatic notation to equations, one often decorates each line with the corresponding index i_j. A scalar, vector, matrix, and a general rank-n tensor are visualized in this notation in figure 2.1.

A *tensor contraction* between two tensors along one or multiple indices is the linear operation that is given by summing over all contracted indices. Given a rank-$(n + f)$

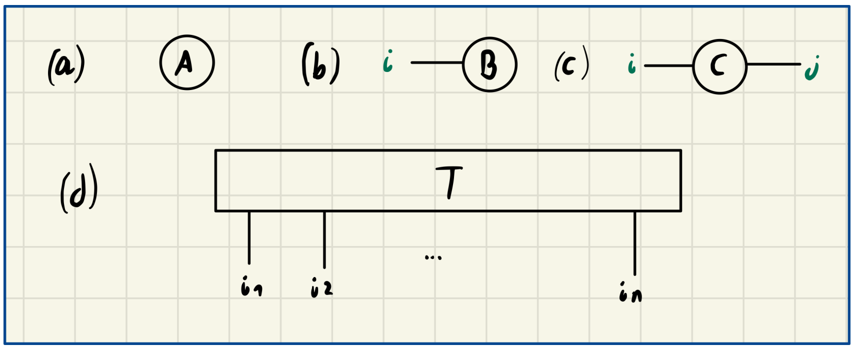

Figure 2.1: Tensors of different ranks are shown in diagrammatic notation. (a) A scalar $a \in \mathbb{C}$. (b) A vector $\boldsymbol{b} \in \mathbb{C}^{\chi}$. (c) A matrix $\boldsymbol{C} \in \mathbb{C}^{\chi_1 \times \chi_2}$. (d) A rank-$n$ tensor $\boldsymbol{T} \in \mathbb{C}^{\chi_1 \times \chi_2 \times \dots \times \chi_n}$

.

tensor $\boldsymbol{X} \in \mathbb{C}^{\chi_1 \times \cdots \times \chi_n \times \xi_1 \times \cdots \times \xi_f}$ and a rank-$(m+f)$ tensor $\boldsymbol{Y} \in \mathbb{C}^{\lambda_1 \times \cdots \times \lambda_m \times \xi_1 \times \cdots \times \xi_f}$, the result of contracting $\boldsymbol{X}$ and $\boldsymbol{Y}$ along the last f indices produces a new rank-$(m+n)$ tensor $\boldsymbol{Z} \in \mathbb{C}^{\chi_1 \times \cdots \times \chi_n \times \lambda_1 \times \cdots \times \lambda_m}$ as

$$Z_{i_1 \ldots i_n j_1 \ldots j_m} \coloneqq \sum_{\alpha_1 = 1}^{\xi_1} \cdots \sum_{\alpha_f}^{\xi_f} X_{i_1 \ldots i_n \alpha_1 \ldots \alpha_f} Y_{j_1 \ldots j_m \alpha_1 \ldots \alpha_f}.$$

Arbitrary contractions can be reformulated as contractions over the last f indices by transposing the tensors. By counting the number of multiplications and additions that are necessary to perform the contraction, the computational complexity can be determined as

$$\mathcal{O}\left(\prod_{\mu=1}^{n} \chi_\mu \prod_{\mu=1}^{m} \lambda_\mu \prod_{\mu=1}^{f} \xi_f \right).$$

A *tensor network* is defined as a collection of tensors that are contracted in a given way. For example, the matrix-vector product of the matrix $\boldsymbol{A} \in \mathbb{C}^{\chi_1 \times \chi_2}$ and the vector $\boldsymbol{b} \in \mathbb{C}^{\chi_2}$ can be written as the contraction of a rank-2 tensor with a rank-1 tensor, resulting in a rank-1 tensor $\boldsymbol{b}' \in \mathbb{C}^{\chi_1}$ with entries

$$b'_i = \sum_{\alpha=1}^{\chi_2} A_{i\alpha} b_\alpha. \tag{2.1}$$

The matrix product of two matrices $\boldsymbol{A} \in \mathbb{C}^{\chi_1 \times \chi_2}$ and $\boldsymbol{B} \in \mathbb{C}^{\chi_2 \times \chi_3}$ can be written as a tensor network of two rank-2 tensors,

$$C_{ij} = \sum_{\alpha=1}^{\chi_2} A_{i\alpha} B_{\alpha j}, \tag{2.2}$$

where the result is another rank-2 tensor $\boldsymbol{C} \in \mathbb{C}^{\chi_1 \times \chi_3}$. As a more involved example we look at a tensor network consisting of two rank-3 tensors $\boldsymbol{A} \in \mathbb{C}^{\chi_1 \times \chi_2 \times \chi_3}$ and $\boldsymbol{B} \in \mathbb{C}^{\chi_2 \times \chi_4 \times \chi_5}$ and one rank-4 tensor $\boldsymbol{C} \in \mathbb{C}^{\chi_3 \times \chi_5 \times \chi_6 \times \chi_7}$, where we contract along the dimensions χ_2, χ_3 and χ_5. The result is a rank-4 tensor $\boldsymbol{D} \in \mathbb{C}^{\chi_1 \times \chi_4 \times \chi_6 \times \chi_7}$:

$$D_{ijkl} = \sum_{\alpha=1}^{\chi_2} \sum_{\beta=1}^{\chi_3} \sum_{\gamma=1}^{\chi_5} A_{i\alpha\beta} B_{\alpha j \gamma} C_{\beta\gamma kl}. \tag{2.3}$$

In tensor network diagrams, contractions are visualized by connecting the lines corresponding to contracted indices. In figure 2.2 we show tensor network diagrams for the tensor networks (2.1), (2.2) and (2.3).
Because tensor contractions are linear, the order in which tensors are contracted doesn't change the result. However, the computational complexity does in general

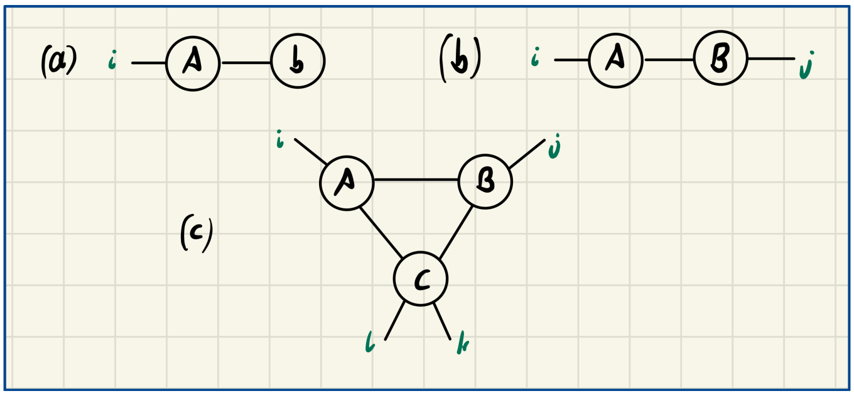

Figure 2.2: Different simple tensor networks are shown in diagrammatic notation. (a) matrix-vector product (2.1). (b) matrix-matrix product (2.2). (c) Tensor network consisting of three tensors (2.3).

depend on the order of contractions and can thus be minimized by choosing the optimal contraction order.
Given two normed vector spaces V_1 and V_2 with $\dim(V_1) = m$, $\dim(V_2) = n$, $m \leq n$, an *isometry* (sometimes also called *semi-unitary*) is a linear, norm-preserving map $W : V_1 \to V_2$ from the smaller to the larger vector space. Each isometry can be represented by a $n \times m$ matrix $\boldsymbol{W}$ fulfilling the *isometry condition*

$$\boldsymbol{W}^\dagger \boldsymbol{W} = \mathbb{1}, \quad \boldsymbol{W}\boldsymbol{W}^\dagger = \mathbb{P}, \tag{2.4}$$

where $\mathbb{P} = \mathbb{P}^2$ is a projection. If $m = n$, it holds $\mathbb{P} = \mathbb{1}$ and W is a *unitary map*. An isometry tensor is a tensor that through grouping of indices and reshaping (i.e. matricization) becomes an isometry. In tensor network diagrams, we draw isometries by decorating lines with arrows. Following [**cite:isometric˙tensor˙network˙states˙in˙two˙dimensions**, **cite:efficient˙simulation˙of˙dynamics˙in˙two˙dimensional˙quantum˙spin˙systems**], we denote the indices belonging to the larger vector space by incoming arrows and the indices belonging to the smaller vector space by outgoing arrows. Unitary tensors are decorated with bidirectional arrows on all indices, where the grouping must be inferred from the context. Ordinary tensors are drawn without arrows. Tensor diagrams for isometric and unitary tensors are shown in figure 2.3.
We lastly introduce an inner product for rank-n tensors $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{C}^{\chi_1 \times \cdots \times \chi_n}$, the *Frobenius inner product*

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle_{\mathrm{F}} \coloneqq \sum_{\alpha_1=1}^{\chi_1} \cdots \sum_{\alpha_n=1}^{\chi_n} A^*_{\alpha_1 \dots \alpha_n} B_{\alpha_1 \dots \alpha_n} = \mathrm{Tr}\left(\boldsymbol{A}^\dagger \boldsymbol{B}\right),$$

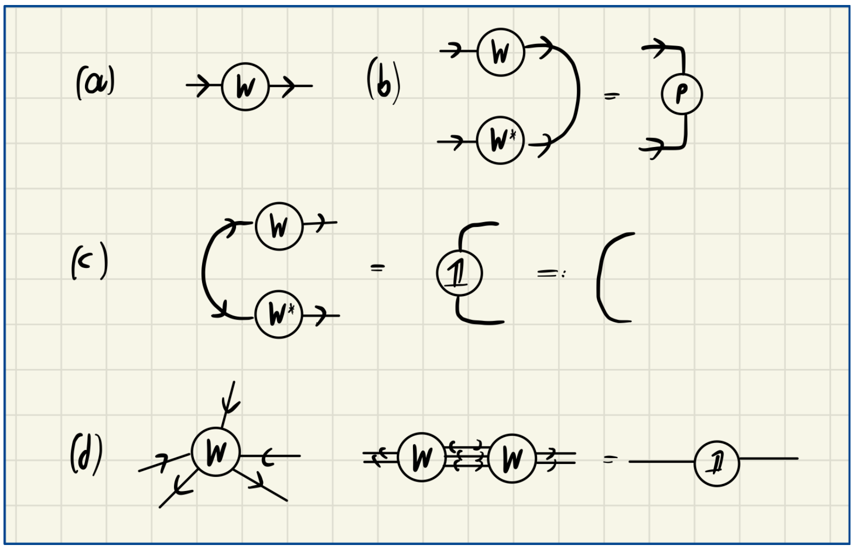

Figure 2.3: Isometric and unitary tensors are drawn by decorating indices with arrows. (a) diagrammatic notation of an isometric matrix (b) (c) The isometry condition (2.4) is depicted diagrammatically. (d) Isomet[illegible]ensors of higher rank must fulfill the isometry condition by grouping of indices.

where the last equality holds only if $n = 2$. The Frobenius inner product induces a norm, the *Frobenius norm*

$$\|\boldsymbol{A}\|_\mathrm{F} = \sqrt{\langle \boldsymbol{A}, \boldsymbol{A}\rangle_\mathrm{F}},$$

which can also be used to define a measure of distance $\|\boldsymbol{A} - \boldsymbol{A}\|_\mathrm{F}$ between tensors $\boldsymbol{A}$ and $\boldsymbol{B}$.

2.2 Tensor Decompositions

There are three decompositions that are used extensively in this thesis: The QR-decomposition, the Singular Value Decomposition, and the Polar Decomposition. All three decompositions are matrix decompositions but can be applied to tensors as well by first grouping indices and reshaping to a matrix, applying the decomposition, and reshaping the result back to the original bond dimensions.
The *reduced QR-decomposition* of a matrix $\boldsymbol{A} \in \mathbb{C}^{n\times m}$ is the decomposition

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}, \tag{2.5}$$

where $\boldsymbol{Q} \in \mathbb{C}^{n\times k}$ is an isometry, $\boldsymbol{R} \in \mathbb{C}^{k\times m}$ is an upper triangular matrix and

$k := \min(n, m)$. The computational complexity of the QR decomposition scales as

$$\mathcal{O}\left(n \cdot m \cdot \min(n, m)\right). \tag{2.6}$$

A diagrammatic depiction of the QR decomposition (2.5) is drawn in figure 2.4(a).
The *Singular Value Decomposition* (SVD) of a matrix $\boldsymbol{A} \in \mathbb{C}^{n\times m}$ is the decomposition

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\dagger, \tag{2.7}$$

where $\boldsymbol{U} \in \mathbb{C}^{n\times k}$ and $\boldsymbol{V} \in \mathbb{C}^{m\times k}$ are isometries, $\boldsymbol{S} \in \mathbb{R}^{k\times k}$ is a diagonal real matrix of *singular values*, and $k := \min(n, m)$. The computational complexity of the SVD is the same as for the QR decomposition (2.6). However, while the scaling is the same, the prefactors are lower for the QR decomposition in most implementations, meaning that the QR decomposition is faster in practice. Moreover, in contrast to the SVD, the QR decomposition allows for highly efficient implementations on graphics processing units (GPUs), which enables decompositions of large matrices to be carried out significantly faster and more power efficiently. Thus, whenever the singular values are not needed, the QR decomposition is preferred over the SVD. Figure 2.4 shows a tensor network diagram of the SVD (2.7).
An important property of the SVD is that it can be used to approximate a matrix $\boldsymbol{A}$ by a matrix $\tilde{\boldsymbol{A}}$ of lower rank $\chi < \min(m, n)$. This *truncated SVD* can be performed by keeping only the largest $\chi < k$ singular values and omitting the corresponding columns of $\boldsymbol{U}$ and $\boldsymbol{V}$:

$$\boldsymbol{A} \approx \tilde{\boldsymbol{A}} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}},$$

with isometries $\tilde{\boldsymbol{U}} \in \mathbb{C}^{n\times\chi}$, $\tilde{\boldsymbol{V}} \in \mathbb{C}^{m\times\chi}$ and real diagonal matrix $\tilde{\boldsymbol{S}} \in \mathbb{C}^{\chi\times\chi}$. It can be shown [**cite:eckart˙young˙theorem**] that the truncated SVD minimizes the distance $\|\boldsymbol{A} - \tilde{\boldsymbol{A}}\|_\mathrm{F}$ between $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}}$ under the constraint $\mathrm{rank}(\tilde{\boldsymbol{A}}) = \chi$. The truncated SVD is frequently used in tensor network algorithms to truncate tensors to a maximum bond dimension χ_max.
The *polar decomposition* of a matrix $\boldsymbol{A} \in \mathbb{C}^{n\times m}$ is the decomposition

$$\boldsymbol{A} = \boldsymbol{W}\boldsymbol{P}, \tag{2.8}$$

where $\boldsymbol{W} \in \mathbb{C}^{m\times n}$ is an isometry and $\boldsymbol{P} \in \mathbb{C}^{n\times n}$ is positive-definite and hermitean. The polar decomposition is related to the SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}$ by

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{V}^\dagger, \quad \boldsymbol{P} = \boldsymbol{V}\boldsymbol{S}\boldsymbol{V}^\dagger.$$

The computational complexity of the polar decomposition is the same as for the QR decomposition and SVD (2.6). The polar decomposition (2.8) is depicted diagrammatically in figure 2.4 One can show that the $\boldsymbol{W}$ factor of the polar decomposition is the isometry "closest" to the matrix $\boldsymbol{A}$, i.e. the isometry that minimizes the distance $\|\boldsymbol{A} - \boldsymbol{W}\|_\mathrm{F}$. Thus, the polar decomposition is often used in isometric tensor network algorithms to "isometrize" tensors.

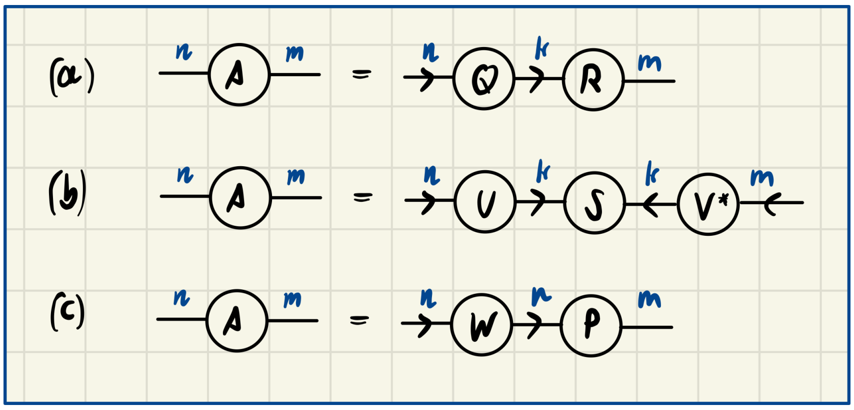

Figure 2.4: Different tensor decompositions are shown in tensor network diagram notation. The indices are decorated with bond dimensions. (a) QR decomposition (2.5). (b) Singular Value Decomposition (2.7). (c) Polar decomposition (2.8).

2.3 Isometric Tensor Networks

An isometric tensor network is a tensor network whose tensor diagrams bonds can be consistently assigned with arrows. In particular we will look at finite tensor networks where the arrows do not form any loops. In such networks, all arrows point to a single tensor, the *orthogonality center*. Such networks have the very useful property, that the error of local approximations around the orthogonality center can be computed locally (without contracting the full network). Let $\mathcal{N}$ be the tensor that is the result of contracting the full network, and let $\mathcal{M}$ be the tensor resulting from the contraction of a subregion of the network around the orthogonality center, where all arrows in the tensor network diagram point towards $\mathcal{M}$ (see figure 2.5(a) for an example in tensor diagram notation). Let us now approximate the sub-network $\mathcal{M}$ by a different sub-network $\mathcal{M}'$, which changes the contraction of the full network to $\mathcal{N}'$ (see 2.5(b)). We can compute the error ϵ of this approximation as

$$\begin{aligned}
\epsilon^2 &= \|\mathcal{N} - \mathcal{N}'\|_\mathrm{F}^2 \\
&= \langle \mathcal{N} - \mathcal{N}', \mathcal{N} - \mathcal{N}' \rangle_\mathrm{F} \\
&= \|\mathcal{N}\|_\mathrm{F} + \|\mathcal{N}'\|_\mathrm{F} - 2\,\mathrm{Re}\,\langle \mathcal{N}, \mathcal{N}' \rangle_\mathrm{F} \\
&= \|\mathcal{M}\|_\mathrm{F} + \|\mathcal{M}'\|_\mathrm{F} - 2\,\mathrm{Re}\,\langle \mathcal{M}, \mathcal{M}' \rangle_\mathrm{F} \\
&= \|\mathcal{M} - \mathcal{M}'\|_\mathrm{F}^2,
\end{aligned}$$

where in the fourth step we used the fact that all tensors outside of the sub-network satisfy the isometry condition. As an example, the contraction of $\mathrm{Tr}\left(\mathcal{N}\mathcal{N}'^\dagger\right)$ is shown in figure 2.5(c). As one can see, the computation of the error reduces to a

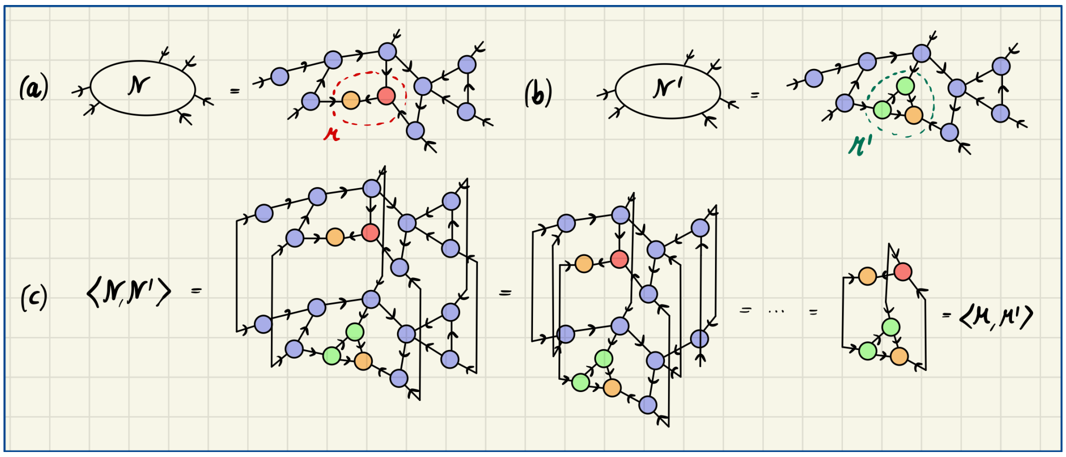
(a)
$\mathcal{N}$
=
$\mathcal{M}$
(b)
$\mathcal{N}'$
=
$\mathcal{M}'$
(c)
$\langle \mathcal{N}, \mathcal{N}' \rangle$ =
=
= ⋯ =
= $\langle \mathcal{M}, \mathcal{M}' \rangle$

Figure 2.5: (a) An isometric tensor network $\mathcal{N}$ with the orthogonality center depicted in orange. The sub-network $\mathcal{M}$ is made up of all tensors in the red region. (b) The isometric tensor network $\mathcal{N}'$ with an updated sub-network $\mathcal{M}'$. (c) The computation of the trace $\mathrm{Tr}\,(\mathcal{N}\mathcal{N}')$ reduces to a contraction of the subregions $\mathrm{Tr}\,(\mathcal{M}\mathcal{M}')$ because of the isometry condition.

contraction of the sub-networks. This greatly simplifies the computation of optimal approximations of tensors, because the full network doesn't need to be contracted. When the tensor network represents a quantum state, this also makes it very easy to compute local expectation values, because the computation of the overlap can be simplified to a contraction of a local environment around the orthogonality center.

2.4 Matrix Product States (MPS)

2.5 Isometric Tensor Product States in 2D

Chapter 3

Isometric Diagonal Tensor Networks (isoDTPS)

3.1 Network Structure

3.2 Yang-Baxter Move

3.3 Time Evolving Block Decimation (TEBD)

Chapter 4

Toric Code: An exactly representable Model

Chapter 5

Transverse Field Ising Model: Ground State Search and Time Evolution

Appendix A

Riemannian Optimization of Isometries

In this appendix we provide a brief introduction to the problem of optimizing a cost function on the constrained set of isometric matrices. This problem can be solved by performing Riemannian Optimization on the matrix manifold of isometric matrices, which is called the Stiefel manifold. For a more in-depth introduction to the topic we recommend the excellent book [1]. A discussion of Riemannian optimization of complex matrix manifolds in the context of quantum physics and isometric tensor networks can be found at [2, 3]. An implementation of Riemannian Optimization on the real Stiefel manifold and other matrix manifolds in python is given in [4]. Some parts of this implementation were also used in our implementation.

A.1 The complex Stiefel manifold

We define the *complex Stiefel manifold* $\mathrm{St}(n,p)$ with $n \geq p$ as the set of all isometric $n \times p$ matrices:

$$\mathrm{St}(n,p) := \left\{ X \in \mathbb{C}^{n\times p} : X^\dagger X = \mathbb{1} \right\}.$$

In particular, for $n = p$, the complex Stiefel manifold reduces to the set of unitary matrices $U(n)$. One can show, similar to [1], that the complex Stiefel manifold is naturally an embedded submanifold of the Euclidian vector space $\mathbb{C}^{n\times p} \cong \mathbb{R}^{2np}$ of general complex $n \times p$ matrices.

Tangent vectors on manifolds generalize the notion of directional derivatives. A mathematical definition of tangent vectors and tangent spaces of manifolds is given in [1]. The set of all tangent vectors to a point $X \in \mathrm{St}(n,p)$ is called the *tangent space* $T_X\mathrm{St}(n,p)$, which is given by [1, 3]

$$T_X\mathrm{St}(n,p) = \left\{ Z \in \mathbb{C}^{n\times p} : X^\dagger Z + Z^\dagger X = 0 \right\}.$$

An arbitrary element $\xi \in \mathbb{C}^{n\times p}$ from the embedding space $\mathbb{C}^{n\times p}$ can be projected to the tangent space $T_X\mathrm{St}(n,p)$ by [1, 3]

$$P_X\xi = \xi - \frac{1}{2}X\left(X^\dagger \xi + \xi^\dagger X\right). \tag{A.1}$$

Additionally, we will also need to define a notion of length that we can apply to tangent vectors. This can be done in the form of an *inner product* on tangent spaces, called the *Riemannian metric.* A natural metric for the tangent space $T_X \mathrm{St}(n,p)$ of the Stiefel manifold is the Euclidean metric of the embedding space $\mathbb{C}^{n\times p}$, which is given by the real part of the Frobenius inner product:

$$g_W : T_X\mathrm{St}(n,p) \times T_X\mathrm{St}(n,p) \to \mathbb{R}, \quad g_X(\xi_1,\xi_2) = \mathrm{Re}\,\mathrm{Tr}\left(\xi_1^\dagger \xi_2\right). \tag{A.2}$$

Equipped with a Riemannian metric the Stiefel manifold becomes a Riemannian submanifold of $\mathbb{C}^{n\times p}$.
With these definition, we can now formulate the optimization problem as the problem of finding the isometry $W_{\mathrm{opt}} \in \mathrm{St}(n,p)$ that minimizes the cost function

$$f : \mathrm{St}(n,p) \to \mathbb{R}, \quad X \mapsto f(X). \tag{A.3}$$

A.2 Gradients, retractions, and vector transport

First order optimization algorihms like Gradient Descent and Conjugate Gradients use the gradient of the cost function to update the search direction at each iteration. In the case of the Stiefel manifold and the cost function (A.3), we first define the matrix of partial derivatives $D \in \mathbb{C}^{n\times p}$ of f at $X \in \mathrm{St}(n,p)$ by

$$D_{ij} := \left.\frac{\partial f}{\partial\,\mathrm{Re}\,(X_{ij})}\right|_X + \mathrm{i}\left.\frac{\partial f}{\partial\,\mathrm{Im}\,(X_{ij})}\right|_X. \tag{A.4}$$

With this definition, the directional derivative $\mathrm{D}f(X)[Z]$ at $X \in \mathrm{St}(n,p)$ in direction $Z \in \mathbb{C}^{n\times p}$ is simply given by an inner product of D with the direction Z, using the Riemannian metric (A.2):

$$\begin{aligned} g_X(D,Z) &= \mathrm{Re}\,\mathrm{Tr}\left(D^\dagger Z\right) = \mathrm{Re}\sum_{ij} D_{ij}^* Z_{ij} \\ &= \sum_{ij}\left(\left.\frac{\partial f}{\partial\,\mathrm{Re}\,(X_{ij})}\right|_X \mathrm{Re}\,Z_{ij} + \left.\frac{\partial f}{\partial\,\mathrm{Im}\,(X_{ij})}\right|_X \mathrm{Im}\,Z_{ij}\right) \\ &=: \mathrm{D}f(X)[Z]. \end{aligned}$$

With this we can now define the gradient $\nabla f(X)$ of f at $X \in \mathrm{St}(n,p)$ as the projection of the partial derivative matrix (A.4) to the tangent space [1, 3]:

$$\nabla f(X) := P_X D = D - \frac{1}{2} X\left(X^\dagger D + D^\dagger X\right),$$

where we used the projection (A.1).

A.3 Conjugate Gradients

A.4 Trust Region Method

Appendix B

Initialization of the Disentangling Unitary

Bibliography

[1] P.-A. Absil, R. Mahony and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton: Princeton University Press, 2008. ISBN: 9781400830244. DOI: doi:10.1515/9781400830244. URL: https://doi.org/10.1515/9781400830244.

[2] Ilia A Luchnikov, Mikhail E Krechetov and Sergey N Filippov. 'Riemannian geometry and automatic differentiation for optimization problems of quantum physics and quantum technologies'. In: *New Journal of Physics* 23.7 (July 2021), p. 073006. ISSN: 1367-2630. DOI: 10.1088/1367-2630/ac0b02. URL: http://dx.doi.org/10.1088/1367-2630/ac0b02.

[3] Markus Hauru, Maarten Van Damme and Jutho Haegeman. 'Riemannian optimization of isometric tensor networks'. In: *SciPost Physics* 10.2 (Feb. 2021). ISSN: 2542-4653. DOI: 10.21468/scipostphys.10.2.040. URL: http://dx.doi.org/10.21468/SciPostPhys.10.2.040.

[4] James Townsend, Niklas Koep and Sebastian Weichwald. *Pymanopt: A Python Toolbox for Optimization on Manifolds using Automatic Differentiation*. 2016. arXiv: 1603.03236 [cs.MS].