



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

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Tensors and Tensor Networks</b>	<b>3</b>
2.1	Conventions and Notation . . . . .	3
2.2	Tensor Decompositions . . . . .	7
2.3	Isometric Tensor Networks . . . . .	8
2.4	Matrix Product States (MPS) . . . . .	9
2.5	Isometric Tensor Product States in 2D . . . . .	11
<b>3</b>	<b>Isometric Diagonal Tensor Networks (isoDTPS)</b>	<b>13</b>
3.1	Network Structure . . . . .	13
3.2	Yang-Baxter Move . . . . .	13
3.3	Time Evolving Block Decimation (TEBD) . . . . .	13
<b>4</b>	<b>Toric Code: An exactly representable Model</b>	<b>15</b>
<b>5</b>	<b>Transverse Field Ising Model: Ground State Search and Time Evolution</b>	<b>17</b>
<b>A</b>	<b>Riemannian Optimization of Isometries</b>	<b>19</b>
A.1	The complex Stiefel manifold . . . . .	19
A.2	Gradients, retractions, and vector transport . . . . .	20
A.3	Conjugate Gradients . . . . .	21
A.4	Trust Region Method . . . . .	21
<b>B</b>	<b>Initialization of the Disentangling Unitary</b>	<b>23</b>
	<b>Bibliography</b>	<b>25</b>



# **Chapter 1**

## **Introduction**





# Chapter 2

## Tensors and Tensor Networks

In the following, a brief introduction to tensors, tensor networks, and tensor network algorithms is given. We start by defining the conventions and notation used in this thesis in section 2.1. In section 2.2, we introduce important tensor decompositions that are used extensively in tensor network algorithms. In section 2.3, we define isometric tensor networks and discuss their properties. Lastly, we give examples for physical states being represented in terms of isometric tensor networks, namely the popular Matrix Product States (MPS) in section 2.4 and the recently developed isometric tensor product states in 2D (isoTPS) in section 2.5.

### 2.1 Conventions and Notation

#### Tensors

For the purpose of this thesis we define a *tensor*  $T$  of rank  $n$  as an  $n$ -dimensional array of complex numbers

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

with entries

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

For example, a rank-0 tensor is a scalar, a rank-1 tensor is a vector, and a tensor of rank-2 is a matrix.

An *index contraction* between two or more tensors is the linear operation that is performed by summing over a given set of indices. For example, the matrix product of two matrices  $A \in \mathbb{C}^{\chi_1 \times \chi_2}$  and  $B \in \mathbb{C}^{\chi_2 \times \chi_3}$  can be written as the index contraction

$$C_{ij} = \sum_{\alpha=1}^{\chi_2} A_{i\alpha} B_{\alpha j}. \quad (2.1)$$

A more involved example is the index contraction of two rank-3 tensors  $A \in \mathbb{C}^{\chi_1 \times \chi_2 \times \chi_3}$ ,  $B \in \mathbb{C}^{\chi_2 \times \chi_4 \times \chi_5}$  and one rank-4 tensor  $C \in \mathbb{C}^{\chi_3 \times \chi_5 \times \chi_6 \times \chi_7}$ , where we

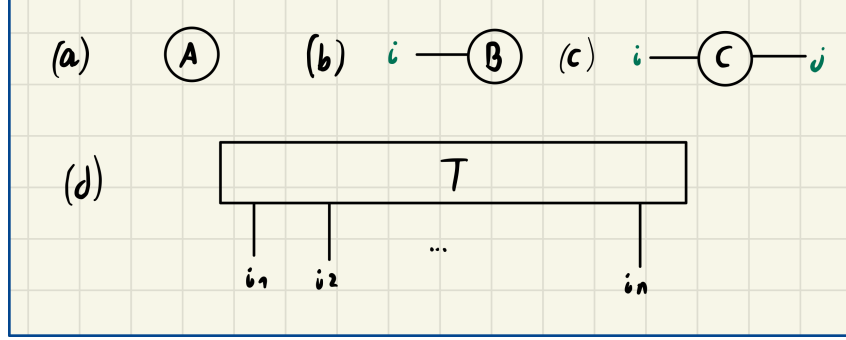


Figure 2.1: Tensors of different ranks are shown in diagrammatic notation.

contract along the indices with dimension  $\chi_2$ ,  $\chi_3$  and  $\chi_5$ . The result is a rank-4 tensor  $D \in \mathbb{C}^{\chi_1 \times \chi_4 \times \chi_6 \times \chi_7}$ :

$$D_{ijkl} = \sum_{\alpha=1}^{\chi_2} \sum_{\beta=1}^{\chi_3} \sum_{\gamma=1}^{\chi_5} A_{i\alpha\beta} B_{\alpha j\gamma} C_{\beta\gamma kl}. \quad (2.2)$$

More generally, given a rank- $(n + f)$  tensor  $A \in \mathbb{C}^{\chi_1 \times \dots \times \chi_n \times \xi_1 \times \dots \times \xi_f}$  and a rank- $(m + f)$  tensor  $B \in \mathbb{C}^{\lambda_1 \times \dots \times \lambda_m \times \xi_1 \times \dots \times \xi_f}$ , the result of contracting  $A$  and  $B$  along the last  $f$  indices produces a new rank- $(m + n)$  tensor  $C \in \mathbb{C}^{\chi_1 \times \dots \times \chi_n \times \lambda_1 \times \dots \times \lambda_m}$  as

$$C_{i_1 \dots i_n j_1 \dots j_m} := \sum_{\mu_1=1}^{\xi_1} \dots \sum_{\mu_f=1}^{\xi_f} A_{i_1 \dots i_n \mu_1 \dots \mu_f} B_{j_1 \dots j_m \mu_1 \dots \mu_f}. \quad (2.3)$$

Contractions over arbitrary indices can be reformulated as contractions over the last  $f$  indices by transposing the tensors. Contractions over more than two tensors can be decomposed into successive contractions of two tensors. Because tensor contractions are linear, the order in which tensors are contracted doesn't change the result. However, the computational complexity does in general depend on the order of contractions and can thus be minimized by choosing the optimal contraction order.

By counting the number of multiplications and additions that are necessary to perform the tensor contraction (2.3), the computational complexity can be determined as

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

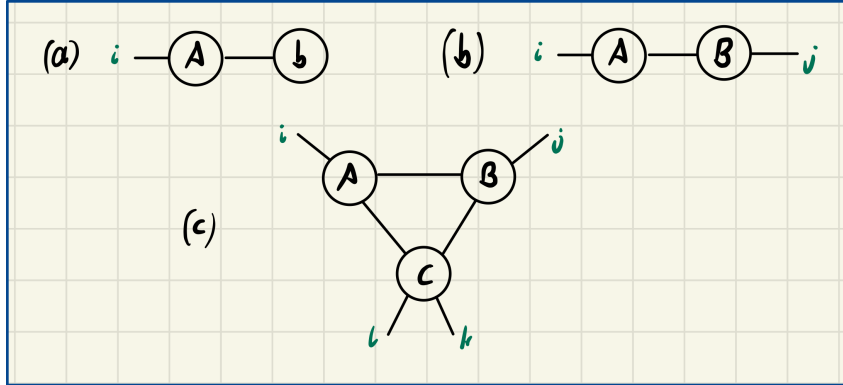


Figure 2.2: Tensor networks in diagrammatic notation. (a) Scalar product (2.4). (b) Matrix product (2.1). (c) More involved network consisting of three tensors (2.2).

### Tensor Networks

A *tensor network* is defined as a collection of tensors that are contracted in a specified way. It is convenient to introduce a diagrammatic notation, where tensors are drawn as shapes and tensor indices are represented by lines emerging from these 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. Index contractions are depicted diagrammatically by connecting the lines corresponding to contracted indices. Lines connecting two tensors are sometimes called *bonds*, while indices not used in contractions are called *open indices*. The *bond dimension*  $\chi_i$  denotes the number of different values an index  $i$  can take. It is often more convenient to discuss tensor network algorithms in terms of diagrams than in terms of equations. Two simple tensor networks are the scalar-product of two vectors

$$c = \sum_{\alpha=1}^{\chi} A_{\alpha} B_{\alpha} \quad (2.4)$$

and the matrix product (2.1). Both networks are shown as diagrams in figure 2.2(a) and 2.2(b) respectively. 2.2(c) depicts the more involved tensor network (2.2).

### Isometries

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  $W : V_1 \rightarrow V_2$  from the smaller to the larger vector space. Each

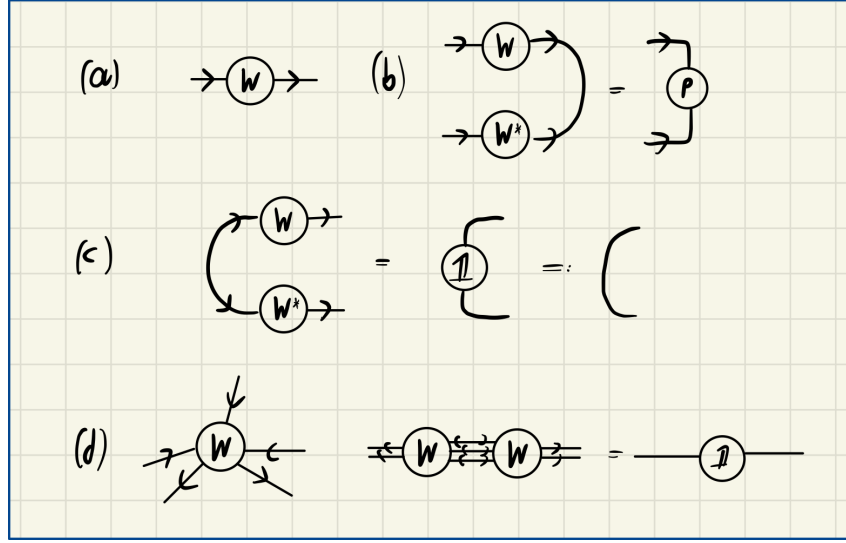


Figure 2.3: Isometric and unitary tensors are drawn by decorating indices with arrows. (a) diagrammatic notation of an isometric matrix (b) (c) The isometry condition (2.5) is depicted diagrammatically. (d) Isometric tensors of higher rank must fulfill the isometry condition by grouping of indices.

isometry can be represented by a  $n \times m$  matrix  $W$  fulfilling the *isometry condition*

$$W^\dagger W = \mathbb{1}, \quad WW^\dagger = \mathbb{P}, \quad (2.5)$$

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 the convention of [1, 2], 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  $A, B \in \mathbb{C}^{\chi_1 \times \dots \times \chi_n}$ , the *Frobenius inner product*

$$\langle A, B \rangle_F := \sum_{\mu_1=1}^{\chi_1} \dots \sum_{\mu_n=1}^{\chi_n} A_{\mu_1 \dots \mu_n}^* B_{\mu_1 \dots \mu_n} = \text{Tr} \left( A^\dagger B \right),$$

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

norm, the *Frobenius norm*

$$\|A\|_F = \sqrt{\langle A, A \rangle_F},$$

which can be used to define a measure of distance  $\|A - B\|_F$  between tensors  $A$  and  $B$ .

## 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  $A \in \mathbb{C}^{n \times m}$  is the decomposition

$$A = QR, \quad (2.6)$$

where  $Q \in \mathbb{C}^{n \times k}$  is an isometry,  $R \in \mathbb{C}^{k \times m}$  is an upper triangular matrix and  $k := \min(n, m)$ . The computational complexity of the QR decomposition scales as

$$\mathcal{O}(n \cdot m \cdot \min(n, m)). \quad (2.7)$$

A diagrammatic depiction of the QR decomposition (2.6) is drawn in figure 2.4(a). The *Singular Value Decomposition* (SVD) of a matrix  $A \in \mathbb{C}^{n \times m}$  is the decomposition

$$A = USV^\dagger, \quad (2.8)$$

where  $U \in \mathbb{C}^{n \times k}$  and  $V \in \mathbb{C}^{m \times k}$  are isometries,  $S \in \mathbb{R}^{k \times k}$  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 (2.7). 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 (2.8).

An important property of the SVD is that it can be used to approximate a matrix  $A$  by a matrix  $\tilde{A}$  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  $U$  and  $V$ :

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

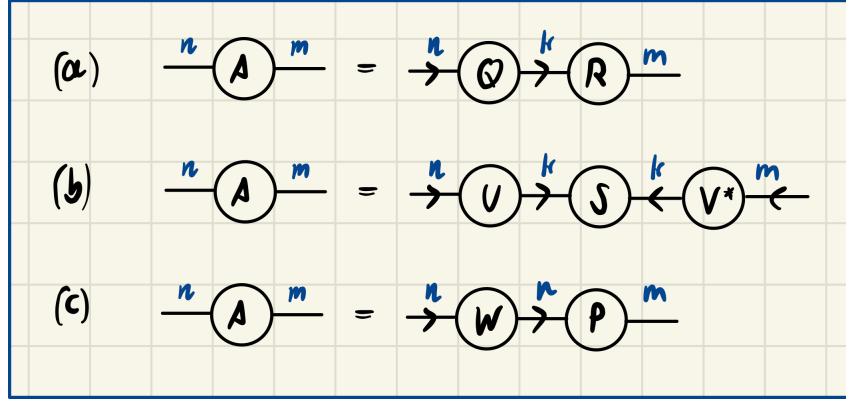


Figure 2.4: Different tensor decompositions are shown in tensor network diagram notation. The indices are decorated with bond dimensions. (a) QR decomposition (2.6). (b) Singular Value Decomposition (2.8). (c) Polar decomposition (2.9).

with isometries  $\tilde{U} \in \mathbb{C}^{n \times \chi}$ ,  $\tilde{V} \in \mathbb{C}^{m \times \chi}$  and real diagonal matrix  $\tilde{S} \in \mathbb{C}^{\chi \times \chi}$ . It can be shown [3] that the truncated SVD minimizes the distance  $\|A - \tilde{A}\|_F$  between  $A$  and  $\tilde{A}$  under the constraint  $\text{rank}(\tilde{A}) = \chi$ . The truncated SVD is frequently used in tensor network algorithms to truncate tensors to a maximum bond dimension  $\chi_{\max}$ . The *polar decomposition* of a matrix  $A \in \mathbb{C}^{n \times m}$  is the decomposition

$$A = WP, \quad (2.9)$$

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

$$W = UV^\dagger, \quad P = VSV^\dagger.$$

The computational complexity of the polar decomposition is the same as for the QR decomposition and SVD (2.7). The polar decomposition (2.9) is depicted diagrammatically in figure 2.4. One can show that the  $W$  factor of the polar decomposition minimizes the distance  $\|A - W\|_F$ . Thus, the polar decomposition is often used in isometric tensor network algorithms to "isometrize" tensors.

## 2.3 Isometric Tensor Networks

An isometric tensor network is a tensor network whose 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  $\epsilon$  of this approximation as

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

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  $\operatorname{Tr}(\mathcal{N}\mathcal{N}^\dagger)$  is shown in figure 2.5(c). As one can see, the computation of the error reduces to a contraction of the sub-networks. This greatly simplifies the computation of optimal approximations of tensors especially for large networks, 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 of the wavefunction can be simplified to a contraction of a local environment around the orthogonality center. Additionally, approximations made by the truncated SVD 2.2 and the isometrization using the polar decomposition 2.9 are, when performed at the orthogonality center, globally optimal for isometric tensor networks, instead of only locally optimal for non-isometric tensor networks.

## 2.4 Matrix Product States (MPS)

Matrix Product States (MPS) have been very successful in solving a variety of problems in one- and two-dimensional lattices. Applications include finding ground and thermal states, real time evolution, and computation of dynamical properties of lattice Hamiltonians. In the following we give a brief introduction to MPS, for a more in-depth discussion see [4–6].

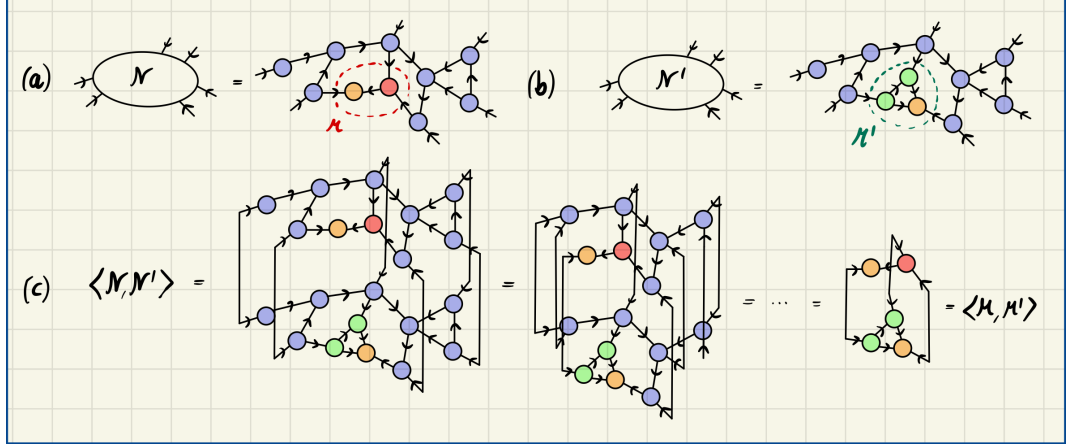


Figure 2.5: (a) An isometric tensor network  $\mathcal{N}$  with the orthogonality center depicted in orange. The sub-network  $\mathcal{M}$  is made up of all tensors in the red region. (b) The isometric tensor network  $\mathcal{N}'$  with an updated sub-network  $\mathcal{M}'$ . (c) The computation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

The state of a quantum many-body system can be written as

$$|\Psi\rangle = \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \cdots \sum_{i_N=1}^{d_N} \Psi_{i_1 i_2 \dots i_N} |i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_N\rangle.$$

where  $N$  is the number of subsystems (e.g. lattice sites or particles), and  $\{|i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_N\rangle\}$  is a set of basis vectors of the full many-body Hilbert space

$$\mathcal{H} = \bigotimes_{j=1}^N \mathcal{H}_j,$$

with  $\dim(\mathcal{H}_j) = d_j$  the dimension of the local Hilbert space of subsystem  $j$ . To simplify the notation, we will assume that the dimension of all local subsystems is the same,  $d_j = d$ . However, the following can easily be generalized to subsystems of different dimensions. The  $d^N$  complex numbers  $\Psi_{i_1 i_2 \dots i_N}$  fully describe the quantum many-body state, and one can think of  $\Psi \in \mathbb{C}^{d \times \cdots \times d}$  as a tensor of rank  $N$ . However, due to the size of the tensor scaling exponentially with system size, only very small system sizes are accessible computationally. One can proceed by writing  $\Psi$  by a tensor network of smaller tensors. A *Matrix Product State* (MPS) does this by



introducing  $N$  rank-3 tensors  $A^{[n]} \in \mathbb{C}^{d \times \chi_{n-1} \times \chi_n}$  and contracting them in a chain as

$$\Psi_{i_1 i_2 \dots i_N} := \sum_{\alpha_1=1}^{\chi_1} \sum_{\alpha_2=1}^{\chi_2} \dots \sum_{\alpha_{N-1}=1}^{\chi_{N-1}} A_{i_1 1 \alpha_1}^{[1]} A_{i_2 \alpha_1 \alpha_2}^{[1]} \dots A_{i_N \alpha_{N-1} 1}^{[N]},$$

## 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 [7]. A discussion of Riemannian optimization of complex matrix manifolds in the context of quantum physics and isometric tensor networks can be found at [8, 9]. An implementation of Riemannian Optimization on the real Stiefel manifold and other matrix manifolds in python is given in [10]. Some parts of this implementation were also used in our implementation.

### A.1 The complex Stiefel manifold

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

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

In particular, for  $n = p$ , the complex Stiefel manifold reduces to the set of unitary matrices  $U(n)$ . One can show, similar to [7], 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 [7]. The set of all tangent vectors to a point  $X \in \text{St}(n, p)$  is called the *tangent space*  $T_X \text{St}(n, p)$ , which is given by [7, 9]

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

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  $T_X \text{St}(n, p)$  by [7, 9]

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

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  $T_X \text{St}(n, p)$  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:

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

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  $W_{\text{opt}} \in \text{St}(n, p)$  that minimizes the cost function

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

## A.2 Gradients, retractions, and vector transport

First order optimization algorithms 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  $X \in \text{St}(n, p)$  by

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

With this definition, the directional derivative  $Df(X)[Z]$  at  $X \in \text{St}(n, p)$  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):

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

With this we can now define the gradient  $\nabla f(X)$  of  $f$  at  $X \in \text{St}(n, p)$  as the projection of the partial derivative matrix (A.4) to the tangent space [7, 9]:

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

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] Michael P. Zaletel and Frank Pollmann. ‘Isometric Tensor Network States in Two Dimensions’. In: *Phys. Rev. Lett.* 124 (3 Jan. 2020), p. 037201. DOI: 10.1103/PhysRevLett.124.037201. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.124.037201>.
- [2] Sheng-Hsuan Lin, Michael P. Zaletel and Frank Pollmann. ‘Efficient simulation of dynamics in two-dimensional quantum spin systems with isometric tensor networks’. In: *Phys. Rev. B* 106 (24 Dec. 2022), p. 245102. DOI: 10.1103/PhysRevB.106.245102. URL: <https://link.aps.org/doi/10.1103/PhysRevB.106.245102>.
- [3] Carl Eckart and Gale Young. ‘The approximation of one matrix by another of lower rank’. In: *Psychometrika* 1.3 (1936), pp. 211–218.
- [4] Ulrich Schollwöck. ‘The density-matrix renormalization group in the age of matrix product states’. In: *Annals of physics* 326.1 (2011), pp. 96–192.
- [5] Román Orús. ‘A practical introduction to tensor networks: Matrix product states and projected entangled pair states’. In: *Annals of physics* 349 (2014), pp. 117–158.
- [6] Johannes Hauschild and Frank Pollmann. ‘Efficient numerical simulations with Tensor Networks: Tensor Network Python (TeNPy)’. In: *SciPost Phys. Lect. Notes* (2018), p. 5. DOI: 10.21468/SciPostPhysLectNotes.5. URL: <https://scipost.org/10.21468/SciPostPhysLectNotes.5>.
- [7] 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>.
- [8] 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>.

- [9] 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>.
- [10] James Townsend, Niklas Koep and Sebastian Weichwald. *Pymanopt: A Python Toolbox for Optimization on Manifolds using Automatic Differentiation*. 2016. arXiv: 1603.03236 [cs.MS].