Numerical bounds on few body sector-lengths

Bachelor thesis by Maximilian Rüsch
Date of submission: November 22, 2023

Supervisor: Prof. Dr. Mariami Gachechiladze
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department

Quantum Computing Group

For science.

Abstract

Analyzing the correlation structure of quantum systems is central to understanding their behaviour. The notion of sector lengths is useful for quantifying correlations in multipartite quantum systems in a basis-independent manner. A complete characterization for sector lengths for two and three qubits exists, however for more qubits this proves difficult to obtain. Instead of deriving properties for four qubits and above, we numerically investigate such systems by obtaining lower bounds on the maximal k-partite correlations for certain $k > 3$. For this purpose, tools from differential geometry are introduced to optimize over arbitrary pure quantum states and certain subsets such as symmetric states. Additionally, we numerically investigate and support a given conjecture of a threshold n_0 after which correlations are upper bounded by $\binom{n}{k}$. We obtain results for the investigation of this threshold n_0 and if it can be discovered using the subsets of pure states.

Acknowledgments

I want to thank Jan Nöller and Mariami Gachechiladze for useful discussions and great supervision during the development of this thesis. Additionally, I thank Nikolai Wyderka for providing insightful material that aided in validating results obtained in this work.

I want to acknowledge the computing time provided on the high-performance computer Lichtenberg at the NHR Centers NHR4CES at TU Darmstadt. This is funded by the Federal Ministry of Education and Research, and the state governments participating on the basis of the resolutions of the GWK for national high performance computing at universities[1].

I thank Fabian Damken for helpful technical infrastructure that smoothed the path of writing this thesis.

Lastly, I thank Becca, Simge, and Thorben for their constructive criticism on this work and their innumerable ways of lightening the mood when most needed.

[1] For more information on funding partners see: www.nhr-verein.de/unsere-partner

Contents

1 Introduction

Correlations between multiple quantum particles are of major importance when studying and understanding quantum phenomena. However, such correlations are not limitless, as they are subject to restrictions from quantum mechanics. Such restrictions include the monogamy relations placed on entanglement [1]. Understanding the structure and nature of these restrictions helps to explore the limits of quantum correlations and is thus central to studying multipartite quantum systems.

The correlation structure of an n-partite quantum system can be described by the notion of *sector lengths* [2]. For each $k \leq n$, the sector length A_k captures the amount of k-partite correlations in the system. Bounds on correlations expressed via sector lengths are used to prove non-existence of some highly entangled states such as absolutely maximally entangled (AME) states of certain n [3]. Additionally, they can be used to construct entanglement detection criteria [4].

Sector lengths are invariant under local unitary operations as they are a quantification of correlation [5]. They are also expressible in terms of purities of their reduced systems [2]. This relation to purities will heavily benefit computation of sector lengths in terms of computational complexity.

Attainable values for individual sectors A_k with $k \in \{2, 3, n\}$ are already characterized. For $k \in \{2, 3\}$ it is proven that there exists a n_0 such that for $n \geq n_0$ they satisfy the bound $A_k \leq \binom{n}{k}$. For $k > 3$ however, the known properties are insufficient to show such an upper bound. Thus, for these $k > 3$, existence of a n_0 for arbitrary k was only conjectured by Wyderka et al. [6]. Interestingly, for the first few odd $k \leq 11$ an upper bound to the value of n_0 can be found using shadow inequalities [7, 8], see Section 3.2.

In this thesis, we aim to numerically investigate the attainable values for individual sectors for different configurations of (n, k). In particular, we are interested in $k > 3$ to support the existence of n_0 for such k. For this purpose, we provide a method for optimization over arbitrary pure quantum states given some real-valued objective function using concepts from differential geometry. We model the set of pure states and some of its subsets as unit spheres, sometimes with respect to special norms. These spheres are Riemannian manifolds, giving a notion of length, direction and movement on them. The notions defined on the Riemannian manifolds are independent of coordinate systems and scale to any dimensionality of vectors. This makes the sets easily traversable by optimization algorithms. We then adapt the widely used family of adaptive moment estimation methods, also referred to as ADAM algorithms [9], to be compatible with this notion of movement. Using these optimization methods, we are able to run our proposed investigation. In order to improve our optimization methods, we propose and study different restrictions to the available pure states such as focusing only on symmetric states. We also compare our proposed values of n_0 for different n with the upper bounds found by shadow inequalities.

This thesis is organized as follows: First, we introduce required notions of quantum information. With these notions, we define sector lengths and discuss their properties along with different computation methods to obtain them from a given quantum state. These computation methods are then cast into efficiently computable objective functions usable for optimization. Furthermore, required notions from differential geometry are

introduced and used to model the set of all pure states and certain subsets as a traversable manifold. We modify existing algorithms for adaptive gradient based optimization to be compatible with such manifolds and their ambient space. We also discuss the possible simplifications of these objective functions when considering restrictions to the available state space. Moreover, we discuss methods used to automatically differentiate and speed up the objective functions, completing the prerequisites for optimizing with gradient based algorithms. Finally, we apply the optimization techniques to find maximally attainable sector lengths and discuss the impact of any placed restrictions on optimization.

2 Notions of quantum information

This chapter provides a short introduction to the basic notions in quantum information and their mathematical description. The main difference between quantum information and its classical counterpart is that information is not always representable by classical bits. In particular, a piece of information does not have to consist of *either* one of the classical states 0, 1 but can also be some mixture of these classical states. This difference requires vastly different handling in information processing. For more in-depth reading and a general introduction to quantum information we refer to the book by Nielsen and Chuang [10].

Given a complex, d-dimensional Hilbert space $\mathcal{H}$ we can model a quantum mechanical pure state as a normalised element of the space. In this thesis we only work in with *qubits* (as quantum generalizations of the classical *bits*), thus we can assume $\mathcal{H}$ to have $d = 2$ for a single qubit. We use the Dirac notation derived from scalar products which denotes a quantum state by a "ket" $|\psi\rangle \in \mathcal{H}$ and its canonical counterpart by a "bra" $\langle\psi| \in \mathcal{H}^\dagger$. We can express states in any orthonormal basis of $\mathcal{H}$. For our purposes we only use the *computational basis* $|1\rangle, \ldots, |d\rangle$ which is exactly the canonical basis of $\mathcal{H}$. In this basis, we express a quantum state as $|\psi\rangle = \sum_i \alpha_i |i\rangle$. The restriction to normalised states is motivated by understanding measurement of the state $|\psi\rangle$ in the basis as finding the system in the state $|i\rangle$ with probability $|\alpha_i|^2$.

When given two Hilbert spaces $\mathcal{H}_A$ with d_A and $\mathcal{H}_B$ with d_B each modeling a quantum system, we obtain their *composite system* with the tensor product $\mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B$. This composite system is again a Hilbert space with $d = d_A \cdot d_B$, thus when working with n qubits the composite system has $d = 2^n$. Additionally, the two subsystems A, B of a composite state $|\psi\rangle_{AB} \in \mathcal{H}_{AB}$ can be independent of each other. In that case, we can express the state as the tensor product $|\psi\rangle_{AB} = |\psi\rangle_A \otimes |\psi\rangle_B$ and we call it *separable*. Otherwise, i.e. when the two subsystems are not independent of each other we call the state *entangled*.

We can evolve pure states by using unitary operators. The simplest of such unitary operators only act on one system at a time, thus they can neither create nor destroy entanglement. These operators are also called local operators. As an example, we present the unitary Pauli operators on one qubit:

$$\sigma_x = \sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

For our purposes, we also define the identity as the fourth Pauli operator $\sigma_0 = \mathbb{1}$. Unitary operators can be combined with the tensor product just like quantum states. A tensor product of multiple Pauli matrices acts locally on each subsystem. Thus, it is exactly equal to the concatenation of applying each Pauli matrix individually on its respective subsystem.

We can measure a quantum state with respect to some property that is captured by an observable. This observable is modeled by a hermitian operator and its eigenvalues ϵ_i model the possible outcomes of measurement. The post-measurement state $|\phi\rangle_i$ associated with the outcome ϵ_i is modeled by the eigenvector to the eigenvalue ϵ_i. When measuring a pure state $|\psi\rangle$ with the observable M, the expectation value of the measurement is given by $\langle\psi| M |\psi\rangle$. Note that each of the Pauli matrices above is also a hermitian matrix and

can therefore be used as an observable. In particular, the tensor product of multiple hermitian matrices is also hermitian, thus we can obtain Pauli observables for composite states.

When measuring one of the two subsystems of a separable state, say $|\psi\rangle_A$, and throwing away the results (i.e. not looking at the outcome of the measurement), we obtain the remaining state $|\psi\rangle_B$ without modifications. As the state is separable the two subsystems are not inherently "connected" and can thus be measured and handled independently of each other. We refer to this measurement type as the *reduction* of the state $|\psi\rangle$ onto the remaining subsystem B. However, when applying this reduction to an entangled state where the two subsystems are not independent we receive a non-pure state. Instead, the system attains the post-measurement state $|\phi\rangle_i$ with probability $\|\epsilon_i\|^2$. We summarize this by an *ensemble* $\{\|\epsilon_i\|^2 = \lambda_i, |\phi\rangle_i\}$ which is a *classical* mixture of pure quantum states. To model such ensembles efficiently, we need a more general framework: the so-called *mixed states*.

We model a mixed state by a *density operator*, a self adjoint, positive semi definite matrix $\rho \in L(\mathcal{H})$ with $\mathrm{Tr}(\rho) = 1$. The mixed state for an ensemble $\{\lambda_i, |\phi\rangle_i\}$ is expressed by $\rho = \sum_i \lambda_i |\phi_i\rangle \langle\phi_i|$. The $|\phi\rangle_i \in \mathcal{H}$ thus form an eigenbasis of this mixed state ρ. We motivate the restriction $\mathrm{Tr}(\rho) = 1$ by observing that the probabilities in the ensemble must sum exactly to 1, thus requiring that $\sum_i \lambda_i = 1$. Additionally, we motivate the positivity of ρ by noting that the probability of the system resulting in a given state must not be negative, thus necessitating that $\lambda_i \geq 0$. In the special case where $\lambda_1 = 1$, implying $\lambda_i = 0$ for the remaining λ_i, the ensemble associated with ρ contains only a single quantum state, thus we call the state ρ *pure*. We can determine the purity, i.e. how close a mixed state ρ is to being pure by calculating $\mathrm{Tr}(\rho^2)$, where it is pure exactly when $\mathrm{Tr}(\rho^2) = 1$.

Lastly, we generalize the notion of measuring quantum states to the mixed states. A generalized measurement with an observable M is modeled by a set of positive operators $E_1, \ldots, E_k$ associated with certain measurement outcomes $1, \ldots, k$. The probability of measuring the outcome j for a state ρ is captured by $\mathrm{Tr}(\rho E_j)$. Similarly, the expectation value of measurement with an observable M is expressed by $\mathrm{Tr}(\rho M)$. Additionally, as we always measure something all probabilities for the outcomes should add up to one, thus we require that $\sum_j E_j = \mathbb{1}$ holds.

3 Sector lengths

In this chapter, we introduce sector lengths as well as discuss their central properties, yielding a first method of computation. Additionally, we introduce the central conjecture which this thesis is interested in. We also discuss current insufficient analytical approaches to proving the conjecture. These insufficient approaches motivate us to search for different numerical approaches which support the conjecture. Lastly, we summarize and extend more efficient computation methods for sector lengths, empowering the aforementioned numerical approaches.

3.1 Definition and basic properties

We are introducing the central object of this thesis, the *sector length*, by defining it with special correlation operators constructed from Pauli matrices. For a more detailed explanation see [2, 6]. Let ρ be a mixed quantum state for n qubits. Furthermore, let $\mathcal{B}$ be the set of all possible length n enumerations of combinations of the four Pauli operators. Explicitly, $\mathcal{B}$ is defined as

$$\mathcal{B} \coloneqq \bigotimes_{0\leq i\leq n} \sigma^{(i)} \text{ with } \sigma^{(i)} \in \{\mathbb{1}, \sigma_x, \sigma_y, \sigma_z\}.$$

Now let $S \subseteq \{1, \ldots, n\}$ be a subset of all parties in the state and $\mathcal{B}_S \subset \mathcal{B}$ the set of all Pauli combinations which act non-trivially on exactly the parties contained in S. Non-trivial action is defined as acting with any Pauli operator other than $\mathbb{1}$. The S-correlation strength $L_S(\rho)$ of a state ρ is defined as the sum of squared expectation values:

$$L_S(\rho) \coloneqq \sum_{\sigma\in\mathcal{B}_S} \langle\sigma\rangle_\rho^2 = \sum_{\sigma\in\mathcal{B}_S} \mathrm{Tr}[\sigma\rho]^2.$$

Definition 3.1. *For $k \in 0, \ldots, n$, the sector length $A_k(\rho)$ is the sum of $L_S(\rho)$ for all S with $|S| = k$:*

$$A_k(\rho) \coloneqq \sum_{|S|=k} \sum_{\sigma\in\mathcal{B}_S} \mathrm{Tr}[\sigma\rho]^2 = \sum_{\Xi_k} \mathrm{Tr}[\Xi_k\rho]^2 \text{ for all } k \in 0, \ldots, n, \tag{3.2}$$

taking Ξ_k as a Pauli operator from any of the $\mathcal{B}_S$ with $|S| = k$.

Note that as for $k = 0$ we have only $S = \emptyset$ and $\mathcal{B}_\emptyset = \{\mathbb{1}^n\}$, thus $A_0 = 1$ by normalization. The sum of all sector lengths of a state is equal to its purity up to a factor of 2^n. Specifically for pure states we have that $\sum_{k=0}^n A_k = 2^n$ as $\rho = \rho^2$.

Sector lengths are invariant under local unitary transformations [5]. This invariance is useful for our upcoming optimizations as it increases the probability of us finding optimal solutions to our optimization problem discussed in Chapter 4.

The sector lengths of a state are connected to its entanglement properties as they can be used to construct entanglement detection criteria [4]. Due to this connection, shedding light on the structure of sector lengths and finding upper bounds on their attainable values is of significant interest. Any results obtained in some basis are valid for any basis due to the invariance of sector lengths to local unitary transformation.

First, we derive some lower bound on the maximal values attainable for sector lengths:

Remark 3.3. *The maximal value attainable for a sector A_k for a state of n qubits is at least $\binom{n}{k}$ as it can be attained by the fully separable state $A_k(|0\rangle^{\otimes n}) = \binom{n}{k}$ [6].*

We continue with the following observation:

Lemma 3.4 (Lemma 1 from [6])**.** *If for all quantum states ρ of n_0 qubits it holds that $A_k(\rho) \leq \binom{n_0}{k}$, then for all states ρ' of $n \geq n_0$, it holds that $A_k(\rho') \leq \binom{n}{k}$.*

It was already shown that for $k \in \{2, 3\}$ such a n_0 exists, i.e. the lower bound from Remark 3.3 also constitutes an upper bound of we take n sufficiently large. The main goal of this thesis is to numerically investigate and support the following conjecture, regarding the existence of such n_0 for all k:

Conjecture 3.5 (Conjecture 6 from [6])**.** *For all k there exists a n_0 such that for all $n \geq n_0$, $A_k \leq \binom{n}{k}$ holds for every state of n qubits.*

We aim to support Conjecture 3.5 by investigating and suggesting n_0 for different A_k and constructing the tools required for this task. In particular, we provide lower bounds on different n_0 by showing that only below these bounds we find states which violate $A_k \leq \binom{n}{k}$.

Remark 3.6. *Note that sector lengths are convex quantities on the mixed states, thus their maximal values are achieved by pure states [6]. Therefore, when numerically exploring bounds on sector lengths we can use optimization techniques and computation methods tailored towards pure state space and subsets thereof.*

3.2 Shadow inequalities

In addition to these lower bounds on n_0, we can also obtain upper bounds on n_0 in certain cases using *shadow inequalities* [11]. Shadow inequalities are a helpful tool in finding bounds on correlations within a state. As an example, they can be used to find upper bounds on the entanglement found in AME states [8].

We can use shadow inequalities to find tight bounds on n_0 for A_2 and A_3 helping to fully characterize the state space associated with their obtainable values [6]. In this section, we review shadow inequalities and their relation to sector lengths.

Given two positive semi-definite hermitian operators M and N acting on a system of n-qubits, the following holds for all $T \subset \{1, \ldots, n\}$ [7, 11]:

$$\sum_{S \subset \{1,\ldots,n\}} (-1)^{|S \cap \bar{T}|} \operatorname{Tr}[\operatorname{Tr}_{\bar{S}}(M) \operatorname{Tr}_{\bar{S}}(N)] \geq 0.$$

The shadow inequalities $S_k \geq 0$ are obtained by summing over all T with $|T| = k$:

$$S_k \coloneqq \sum_{T,S\subset\{1,\ldots,n\}} (-1)^{|S\cap\bar{T}|} \operatorname{Tr}[\operatorname{Tr}_{\bar{S}}(M) \operatorname{Tr}_{\bar{S}}(N)] \geq 0.$$

We evaluate the inequalities in terms of sector lengths for a specific state $M = N = \rho$ [12, 13]:

$$S_k = \frac{1}{2^n} \sum_{r=0}^{n} (-1)^r K_k(r;n) A_r \geq 0$$

with

$$K_k(r;n) = \sum_{j=0}^{k} (-1)^j 3^{k-j} \binom{r}{j} \binom{n-r}{k-j}.$$

We can add further constraints derived from properties of pure states which are expressible in terms of sector lengths as well [6, 14]:

$$M_m \coloneqq 2^m \sum_{j=0}^{n-m} \binom{n-j}{m} A_j - 2^{n-m} \sum_{j=0}^{m} \binom{n-j}{n-m} A_j = 0 \text{ for } 0 \leq m \leq n.$$

In coding theory, these constraints $M_m = 0$ are known as MacWilliams identities [15].

With the shadow inequalities, MacWilliams identities, and the purity constraint, linear programs can be used to find the upper bound on n_0 for $k = 3$ [6]. Note that the upper bound on n_0 for $k = 2$ can be obtained from the purity constraints alone without the shadow inequalities.

Shadow inequalities can also be obtained for $k \geq 3$. However, they appear to only manage to provide any upper bound on n_0 when k is odd. Additionally, they are insufficient to form a tight upper bound on n_0 for odd $k > 3$. The results for some small k, which we obtained from internal discussions, can be found in Table 6.17.

3.3 Calculating sector lengths from purities

Computing the sector lengths in ways similar to their original definition provided in Theorem 3.2 has the drawback of being very expensive. Enumerating every relevant Pauli operator Ξ_k for a given k necessitates many matrix multiplications and, depending on the implementation, high intermediate storage requirements. This motivates the search for more efficient computation methods which are achieved by computing sector lengths over purities of reductions of the state. This section discusses the relation of purities of reductions of a state to its sector lengths and Section 3.4 concerns leveraging this relation in computations.

Let ρ_S denote the reduction of ρ onto the set S. It is already noted that a given $L_S(\rho)$ will only depend on ρ_S and the purities of its reductions [2]

$$L_S(\rho) = L_S(\rho_S) = \sum_{S'\subset S} (-1)^{|S|-|S'|} d_{S'} \operatorname{Tr}\left[\rho_{S'}^2\right], \tag{3.7}$$

where $d_{S'} = \prod_{a\in S'} d_a$ with d_a being the dimensionality of the system for party a. For our case, $\forall a \in S' : d_a = 2$ holds as we only work with qubits.

This approach is a good improvement upon the original computation for a single $L_S(\rho)$ as we no longer need to compute all $\sigma \in \mathcal{B}_S$. However, when computing A_k and using all $\mathcal{B}_S$ with $|S| = k$, another approach using purities inspired from weight enumerators in coding theory emerges. Weight enumerators are bivariate polynomials which refer to their coefficients as *weights*. If we denote the weights of an enumerator by A_w for $0 \leq w \leq n$ we can construct its general form [16]

$$A(x, y) \coloneqq \sum_{0 \leq w \leq n} A_w x^{n-d} y^d. \tag{3.8}$$

It turns out that sector lengths are a special case of the weights in the *primary Shor-Laflamme weight enumerator* A [16, 17]. Recall that $\mathcal{B}$ is the set of all n-qubit Pauli operators. Let $wt(E)$ be the *weight* of a given $E \in \mathcal{B}$, denoting the number of non-trivially acting Pauli operators in E. Then for two operators M_1, M_2, a single weight $A_d(M_1, M_2)$ in A (disregarding normalization) is defined as

$$A_d(M_1, M_2) \coloneqq \sum_{\substack{E \in \mathcal{E} \\ wt(E)=d}} \mathrm{Tr}(EM_1)\,\mathrm{Tr}(EM_2).$$

For the special case where $M_1 = M_2 = \rho$, this yields

$$A_d(\rho) = \sum_{\substack{E \in \mathcal{E} \\ wt(E)=d}} \mathrm{Tr}(E\rho)^2.$$

Since $\{E \in \mathcal{B} | wt(E) = d\} = \{\Xi\}_d$, we obtain the exact definition of a sector length

$$A_d(\rho) = \sum_{\Xi_d} \mathrm{Tr}(\Xi_d \rho)^2.$$

To summarize, the weight A_d of the primary Shor-Laflamme enumerator (up to normalization) is exactly equal to the sector length A_k with $k = d$. For a given M_1, M_2, the full enumerator polynomial A can be constructed using Theorem 3.8:

$$A(x, y) \coloneqq \sum_{0 \leq d \leq n} A_d(M_1, M_2) x^{n-d} y^d$$

and analogously using $A_d(\rho)$.

Motivated by the primary Shor-Laflamme enumerator, an additional enumerator A' known as the *Rains primary enumerator* can be defined for a set of parties S and two operators M_1, M_2 [16]:

$$A'_S(M_1, M_2) \coloneqq \mathrm{Tr}_S(\mathrm{Tr}_{S^c}(M_1)\,\mathrm{Tr}_{S^c}(M_2))$$

with Tr_S being the partial trace over the parties S and S^c denoting the complement of S. A single weight of the new enumerator is defined as

$$A'_d(M_1, M_2) \coloneqq \sum_{|S|=d} A'_S(M_1, M_2).$$

Setting $M_1 = M_2 = \rho$ again yields the purity of the reduction ρ_S:

$$A'_S(M_1, M_2) = \mathrm{Tr}_S(\mathrm{Tr}_{S^c}(M_1)\,\mathrm{Tr}_{S^c}(M_2)) = \mathrm{Tr}_S(\rho_S^2) = \mathrm{Tr}\big(\rho_S^2\big) =: A'_S(\rho),$$

which reduces the weights of A' to the sum over purities of all reductions of the state onto d parties:

$$A'_d(M_1, M_2) = \sum_{|S|=d} A'_S(\rho) = \sum_{|S|=d} \mathrm{Tr}\left(\rho_S^2\right) =: A'_d(\rho).$$

Defining the full Rains enumerator polynomial A' analogously to A, it is proven in corollary 5 and theorem 7 of [16] that both enumerators are related by:

$$A'(x, y) = A(x + y/2, y/2) \text{ or equivalently } A(x, y) = A'(x - y, 2y). \tag{3.9}$$

The transformation Theorem 3.9 gives us a tool to compute sector lengths of a state ρ by only summing purities of reductions of ρ. We first compute all A'_d and then use the relation of A' to A yielding a polynomial on x, y containing the sector lengths of ρ. To extract these sector lengths we can set $x = 1$ and solve for the coefficients of the remaining y^d:

$$A(x, y) = \sum_{0 \le d \le n} A_d(\rho) x^{n-d} y^d \overset{x:=1}{=} \sum_{0 \le d \le n} A_d(\rho) y^d.$$

3.4 Efficient sector length computation

A single weight A'_d can also be obtained only using A_i with $0 \le i \le d$ [16]:

$$A'_d = 2^{-d} \sum_{0 \le i \le d} \binom{n-i}{n-d} A_i.$$

However, the reverse is also true as inspired by $L_S(\rho)$ from Theorem 3.7 depending only on the purities of the reductions of ρ_S:

Corollary 3.10 (to Theorem 3.9)**.** *For any $k \in 0, \ldots, n$, A_k can be obtained from only A'_d with $0 \le d \le k$, following the identity*

$$A_k = \sum_{0 \le d \le k} (-1)^{k+d} 2^d \binom{n-d}{n-k} A'_d.$$

Proof. Using Theorem 3.9, we begin to solve for A_i:

$$\begin{aligned}
A(x, y) &= \sum_{0 \le d \le n} A'_d (x - y)^{n-d} (2y)^d \\
&= \sum_{0 \le d \le n} A'_d 2^d \sum_{0 \le i \le n-d} \binom{n-d}{i} (-1)^{n-d-i} x^i y^{n-d-i} y^d \\
&= \sum_{0 \le d \le n} A'_d 2^d \sum_{0 \le i \le n-d} \binom{n-d}{i} (-1)^{n-d-i} x^i y^{n-i} \\
&= \sum_{0 \le i \le n} \underbrace{\sum_{0 \le d \le n} A'_d 2^d \binom{n-d}{i} (-1)^{n-d-i}}_{=A_{n-i}} x^i y^{n-i}.
\end{aligned}$$

In the last step, we used that $\binom{n-d}{i} = 0$ for $i > n - d$ to free the summation index i from dependence on d. Setting $k = n - i$ and reversing the summation index reveals our target A_k:

$$\begin{aligned} A_{n-i} &= \sum_{0 \leq d \leq n} A'_d 2^d \binom{n-d}{i} (-1)^{n-d-i} \\ A_k &= \sum_{0 \leq d \leq n} A'_d 2^d \binom{n-d}{n-k} (-1)^{k+d} \\ &= \sum_{0 \leq d \leq k} (-1)^{k+d} 2^d \binom{n-d}{n-k} A'_d. \end{aligned}$$

□

Corollary 3.10 enables us to significantly reduce the number and size of computed reductions of ρ, leading to a considerable improvement in performance of computation.

3.5 Sector length computation for symmetric states

There is a subset of pure states called *symmetric states* which are invariant under exchange of system pairs. For $0 \leq i, j \leq n$, let $\hat{P}_{ij}$ be the unitary operator that swaps the subsystems i and j. Then a n-qubit state ρ_S is a symmetric state exactly when the following holds:

$$\hat{P}_{ij}\rho_S = \rho_S \qquad \forall i, j. \tag{3.11}$$

For pure states we can express this restriction alternatively in the following way: Let $wt_H(a)$ be the Hamming weight of a binary string a, i.e. the number of ones in the string a. Furthermore, take the expression of a state $|\psi\rangle$ in its computational basis:

$$|\psi\rangle = \sum_{a \in \{0,1\}^n} c_a |a\rangle$$

For $|\psi\rangle$ to be a symmetric state, we require that all $|a\rangle$ with the same Hamming weight $wt_H(a)$ share the same coefficient c_a. More formally, we require that for all $a, b \in \{0,1\}^n$ it holds that $wt_H(a) = wt_H(b) \Rightarrow c_a = c_b$.

Consider the set $H_k = \{a | wt_H(a) = k, a \in \{0,1\}^n\}$ of all strings with length n and weight k. The equal superposition of all $|h\rangle$ with $h \in H_k$ is commonly known as the Dicke state $|D^n_k\rangle$ [18]. Expressing any state of n-qubits using Dicke states automatically enforces the symmetry condition from Theorem 3.11. Exchanging systems in such states will only produce permutations of binary strings which share the coefficient, thus staying effectively invariant:

$$\hat{P}_{ij} \sum_{k=0}^{n} c_k |D^n_k\rangle = \sum_{k=0}^{n} c_k \hat{P}_{ij} |D^n_k\rangle = \sum_{k=0}^{n} c_k |D^n_k\rangle .$$

Symmetric states as expressed via Dicke states can be fully described by a greatly reduced number of parameters. A general n-qubit pure state requires (in vector notation) exponentially growing 2^n parameters, while a symmetric state of an equal qubit count only requires linearly growing $n + 1$ parameters. This makes symmetric states easier to handle in computational applications. In particular, derivatives of sector lengths for symmetric states will be easier to compute.

For symmetric states, one can further optimize computations. For a given k, all possible reductions of a state ρ_S to k parties must share the same sector A_k due to the invariance of ρ_S under permutations. With this, one can compute a single reduction ρ_{S_0} and multiply the resulting sector lengths by the number of possible different party sets for reduction:

$$A_k(\rho) = \binom{n}{k} A_k(\rho_{S_0}). \tag{3.12}$$

Lastly, it may be possible to fully avoid computation of reduced density matrices for symmetric states. At least for Dicke states themselves, computations of their S-correlation strength L_S and thus also of their sector lengths A_k are directly possible without computing reduced density matrices. The purity of the reduced systems can be explicitly given due to the high symmetry in the original state [2]. However, it is a lot harder to give a closed form purity expression for the full range of symmetric states as they are linear combinations of Dicke states. As will be clarified in the following chapters, a technique like this may lower computational resource cost significantly and allow exploration of much higher k than currently possible.

4 State Optimization

A common method to optimize in spaces of general quantum states is using semi definite programs (SDPs) as the common representation of states and any operations defined on them work with positive semi-definite matrices [19]. In this sense, SDPs are a natural optimization method for problems in quantum information science. However, SDPs become increasingly harder to implement for non-linear objective functions (like sector lengths) as one needs to introduce auxiliary variables to find some new linear objective function. SDPs often require non-trivial constraints to give any useful results. Finding these constraints can be cumbersome and hard to come up with for e.g. fidelity estimation problems [19]. This requirement of SDPs for linear objective functions and non-trivial constraints stems from semi-definite programming being a subfield of convex optimization. It raises the entry barrier and even limits the range of possible objective functions for use in SDPs.

We want to take another approach of using stochastic gradient-based optimization methods as they are usually more accessible and easier to implement than SDPs. The simplest of these methods is stochastic gradient descent (SGD) but more advanced methods like the adaptive family of ADAM [9] and alike exist. These methods however, in their original definition, are restricted to the Euclidean space $\mathbb{R}^n$. Several works extend the definitions of the methods and most of their guarantees (e.g. with respect to convergence) to a more general domain called *Riemannian manifolds* [20, 21]. This domain from the field of differential geometry enables study and experiments with optimization spaces that are not necessarily Euclidean. We want to combine the simplicity and convergence guarantees of Euclidean ADAM and the flexibility of Riemannian manifolds regarding the modeling of search spaces.

In this chapter, we will model the pure state space and some of its subsets with spheres. We handle these spheres as Riemannian manifolds and use tools defined on those manifolds to define movement from a point with a given direction. Additionally, we adapt the ADAM algorithm to be able to use this movement on the manifold to perform a gradient-based optimization.

4.1 Notes on differential geometry

This section covers some basic concepts of differential geometry required to understand the ideas following in the next sections. For a more comprehensive introduction to the field we refer the reader to Spivak [22] or Robbin and Salamon [23].

Manifolds are a generalization of surfaces in higher dimensions. A manifold $\mathcal{M}$ of dimension n is thus a hypersurface that is embedded in a surrounding Euclidean space $\mathbb{R}^m$ (for spheres it can be fixed with $m = n + 1$) and can be locally approximated in a first-order manner by other Euclidean spaces of dimension n called *tangent spaces*. The tangent space for a specific point $x \in \mathcal{M}$ (which we will refer to as the *anchor* of the tangent space) is expressed by $T_x\mathcal{M}$. A tangent space lets us define its local geometry by adding a

point-specific inner product usually expressed by $g_x(\cdot,\cdot) : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$. The collection of all g_x across the tangent spaces is called the *Riemannian metric* g of a manifold $\mathcal{M}$ which then becomes a Riemannian manifold as the pair $(\mathcal{M}, g)$.

Defining the metric g enables us further to define a norm $\|\cdot\|_x$ local to the anchor $x \in \mathcal{M}$ by $\|v\|_x \coloneqq \sqrt{g_x(v,v)}$ for a tangent vector $v \in T_x\mathcal{M}$. We can express the length $l(c)$ of a curve $c(\cdot) : [a,b] \to \mathcal{M}$ via the norm by integrating over its local speed along the curve: $l(c) \coloneqq \int_{t=0}^{1} \left\| \dot{c(t)} \right\|_{c(t)} dt$. Curves with locally minimal length (a generalization of the concept of a "straight" line) are called *geodesics* and are usually denoted by γ. Note that a geodesic between two points in $\mathcal{M}$ is not necessarily the shortest path between them. An instance of this is found in the generalization of a centered circle on a sphere manifold called a "great circle". Any arc of any great circle is a geodesic. In particular, the two possible arcs connecting the two points in question on a great circle are geodesics. However, only one of them is the globally shortest path between the two points.

As tangent spaces are only local approximations, we cannot take a step from an anchor $x \in \mathcal{M}$ in any direction $v \in T_x\mathcal{M} \subseteq \mathbb{R}^{n+1}$ in the ambient space and still expect to end up on a point on the manifold $\mathcal{M}$. This can be solved (under some additional assumptions which hold for spheres) by the *exponential map* $\exp_x : T_x\mathcal{M} \to \mathcal{M}$ for a given anchor x. The exponential map defines the movement along a geodesic from its given anchor x in an initial direction $v \in T_x\mathcal{M}$. A simple example for this is the exponential map for $\mathbb{R}^n$, which is defined as a simple translation $\exp_x(u) = x + u$. We can depict movement on the manifold for small t in the initial direction v with $\exp_x(tv)$.

Movement from a point x to a point y on the manifold may alter our local geometry (e.g. for any manifold with some curvature like a sphere). Thus, we need to transport any vectors initially defined in $T_x\mathcal{M}$ (e.g. any state produced by optimization methods such as *momentum* in ADAM) without altering their local interpretation. In Euclidean space without curvature this can be easily done via translating a given vector v from x to y, since the local geometry stays the same. For a sphere manifold, simple translation (direction of the vector stays fixed in surrounding $\mathbb{R}^{n+1}$) causes the vector to rotate relative to the surface of the manifold. Notably, the resulting direction of the vector may depend on the taken path from x to y. This is why a special construction $P_x(v; w)$ called *parallel transport* is introduced that can transport a vector v from its anchor x along a geodesic initialized with direction and velocity w. While an exact parallel transport may be computationally expensive, one can use a simple projection onto the next tangent space $T_y\mathcal{M}$ as an approximation for very small movements. This particular approximation on the sphere may degrade the state to be transported by some amount. However, for most applications using adaptive methods this degradation is negligible and can (for very small movements) be disregarded.

4.2 Optimization in pure state space

Recall from Remark 3.6 that the sector length A_i is a convex quantity, thus its maximal values can be attained by pure states. This enables us to confine the search space for any optimization method to the pure states which due to normalization can be naturally understood to be on the unit sphere. Thus, we can use simple movement via the exponential map without leaving pure state space and are guaranteed that the optimal states for any given target quantity are on the sphere. Explicitly, for any number of qubits n, taking $\mathbb{R}^{2^n}$ as the ambient space of a sphere $\mathcal{S}^{2^n-1}$ with $\mathcal{S}^{2^n-1} \coloneqq \{v \in \mathbb{R}^{2^n} : \|v\| = 1\}$ (using the standard Euclidean norm), we can see that $\mathcal{S}^{2^n-1}$ only contains pure quantum states (with no imaginary parts to their parameters). As pure states are usually defined with complex coefficients in some $\mathbb{C}^{2^n}$, we can also define a fitting sphere

$\mathcal{S}^{2^{n+1}-1} \coloneqq \{v \in \mathbb{R}^{2^{n+1}} : \|v\| = 1\}$ which intuitively splits complex parameters into their real and imaginary parts.

Which exact mapping (i.e. parameter splitting method) from $\mathbb{C}^{2^n}$ to $\mathbb{R}^{2^{n+1}}$ is chosen is irrelevant as long as it is kept constant throughout all calculations. We will further refer to $\mathcal{S}^{2^n-1}$ as $\mathcal{S}^n_{P,\mathbb{R}}$ (where P stands for *pure* states) and to $\mathcal{S}^{2^{n+1}-1}$ as $\mathcal{S}^n_{P,\mathbb{C}}$. Any properties holding for $\mathcal{S}^n_{P,\mathbb{C}}$ not explicitly requiring complex numbers also hold for $\mathcal{S}^n_{P,\mathbb{R}}$.

We model these target quantities as objective functions $f : \mathcal{S}^n_{P,\mathbb{C}} \to \mathbb{R}$. Intuitively, this gives the sphere a notion of height by using the value of f at any given point to scale its position vector. The state with $\max_{|\psi\rangle} f(|\psi\rangle)$ can thus be thought of as being represented by the highest point on the sphere (similarly for $\min_{|\psi\rangle} f(|\psi\rangle)$). However, searching the entire state space for these points is highly inefficient and exhaustive searches are outright impossible as there is an infinite number of pure states. A notion of height on a surface levels the ground for the use of gradient-based optimization methods which can be much more efficient at finding optimal states.

It is important to note that whilst the general optimization problem posed by sector lengths is convex we lose this property when focusing on pure states only. Thus, we may get stuck in local extrema, not resulting in the globally optimal state. Reducing the ratio of such suboptimal results is all about choosing the right optimization methods / fitting hyperparameters.

4.3 Optimization in symmetric state space

In this section we consider spheres analogous to the previously defined $\mathcal{S}^n_{P,\cdot}$ for the symmetric states introduced in Section 3.5. As the number of required parameters is significantly reduced for symmetric states, it enables us to visualize the real sphere for $n = 2$, the first case where a $0 \neq k \neq n$ is possible.

On the sphere $\mathcal{S}^n_{P,\mathbb{C}}$ we most likely encounter non-symmetric states. However, for optimization purposes it is helpful to traverse a manifold containing only states from the space that fulfills all restrictions (in this case states being symmetric). Building a naive sphere by normalization $\|v\| = 1$ of a vector $v \in \mathbb{R}^{n+1}$ ($\mathbb{R}^{2(n+1)}$ for complex parameters) will not yield quantum states when attempting reconstruction of symmetric states by multiplication with Dicke states. The coefficient c_k will occur exactly $\binom{n}{k}$ times in the final state vector (as exactly $\binom{n}{k}$ permutations of a n-length string with Hamming Weight k exist), but the existing norm assumes it only does so once.

To work around this problem, we define a new *binomial weighted* "norm" $\|\cdot\|_W$ that takes the occurrences of the different c_k into account. To ensure receiving valid quantum states upon normalization with $\|\cdot\|_W$, we define the normalization by following three steps for a given $v \in \mathbb{R}^{n+1}$:

1. Produce a weighted v_W using a diagonal $n \times n$ matrix M that contains $\binom{n}{k}$ in row / column k
2. Normalize v_W using the regular ambient norm $\|\cdot\|$
3. Regain a non-weighted v' by using M^{-1}

Definition 4.1. *Given the ambient norm $\|\cdot\|$ in a m-dimensional real Hilbert space X, the **binomial weighted norm** $\|v\|_W$ of a vector $v \in X$ is defined as*

$$\|v\|_W \coloneqq \|Mv\| \text{ with } M_{ij} \coloneqq \delta_{ij} \cdot \binom{m}{i} \text{ for } 0 \leq i, j \leq m.$$

*The matrix M is referred to as the **binomial mapping** alongside M^{-1} as the **reverse binomial mapping**.*

Note that $\|\cdot\|_W$ is a valid norm as M is clearly invertible. We then use the new weighted norm $\|\cdot\|_W$ to construct two new spheres for the symmetric states (identified with S) $\mathcal{S}^n_{S,\mathbb{R}}$ and $\mathcal{S}^n_{S,\mathbb{C}}$ analogous to previous constructions.

Geometrically, the norm $\|\cdot\|_W$ just produces a "squished" version of the hypersphere in $\mathbb{R}^{n+1}$ when finding all vectors $v \in \mathbb{R}^{n+1}$ satisfying $\|v\|_W = 1$. As an example, we present the sphere for $n = 2$ in the Figs. 4.2 to 4.5. We visualized the sectors A_1 and A_2 respectively with and without a height mapping to illustrate the distribution of sector lengths on the sphere. It is important to note that there seem to be significant symmetries in the sector length distributions. This aligns with the sector lengths being invariant under local unitary action, thus naturally exhibiting some rotational symmetry.

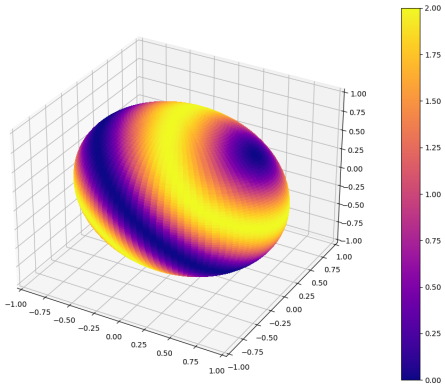

Figure 4.2: Sphere for $\mathcal{S}^2_{S,\mathbb{R}}$ with a visualization of the sector A_1 without a height mapping.

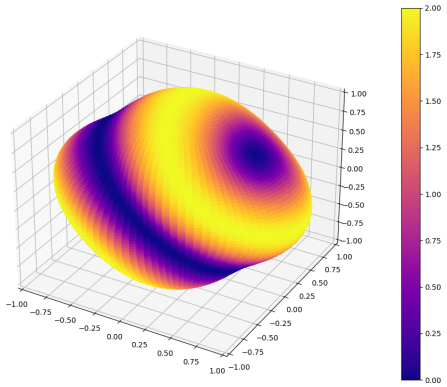

Figure 4.3: Sphere for $\mathcal{S}^2_{S,\mathbb{R}}$ with a visualization of the sector A_1 with a height mapping.

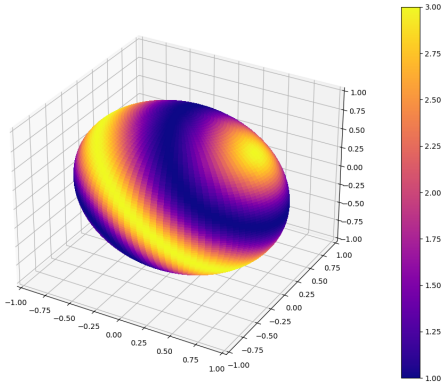

Figure 4.4: Sphere for $\mathcal{S}^2_{S,\mathbb{R}}$ with a visualization of the sector A_2 without a height mapping.

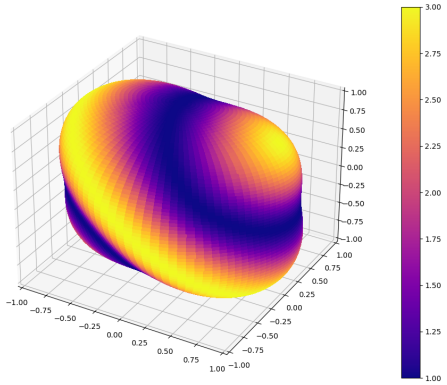

Figure 4.5: Sphere for $\mathcal{S}^2_{S,\mathbb{R}}$ with a visualization of the sector A_2 with a height mapping.

When using complex vectors, they first have to be "collapsed" from their usual sphere in $\mathbb{R}^{2(n+1)}$ to $\mathbb{C}^{n+1}$ in order to be compatible with $\|\cdot\|_W$. After normalization, they can be "expanded" again to $\mathbb{R}^{2(n+1)}$ for the remaining optimization step.

Moving on the new spheres $\mathcal{S}^n_{S,\cdot}$ works differently than discussed for $\mathcal{S}^n_{P,\cdot}$. Since we want to use the geodesics as defined on an "unsquished" sphere (for simplicity) we have to map our positions on $\mathcal{S}^n_{S,\cdot}$ to the hypersphere

defined in the same Euclidean ambient space. For this, we cannot just map our position with the binomial mapping. Using the binomial mapping will in some cases change the rotation of our position vector relative to the obtained gradient we use for stepping (see Section 4.4), as not all dimensions are scaled equally. To be exact, in order to move on the sphere we first need to determine our position on the regular hypersphere by scaling with its regular norm $\|\cdot\|$. Once the regular hypersphere is reached we can move using the same exponential map as on $\mathcal{S}^n_{P,\cdot}$. After movement on the regular hypersphere we determine the target position on the sphere $\mathcal{S}^n_{S,\cdot}$ by scaling with the weighted norm $\|\cdot\|_W$.

4.4 Riemannian Optimization Methods

This section presents the idea of updating the regular ADAM algorithm for usage on the ambient spaces of a given sphere $\mathcal{S}^n_{X,Y}$ to calculate gradient based updates to the current position on the manifold. The regular ADAM algorithm is defined as [9]:

Algorithm 4.6: Original ADAM as defined in [9]. Recommended values: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$

require : α: Stepsize, $\beta_1, \beta_2 \in [0, 1)$: Decay rates for moment estimates, ϵ: Numerical stability additive
require : $f(\theta)$: Objective function on parameters θ, θ_0: Initial parameter vector

$m_0 \leftarrow 0$
$v_0 \leftarrow 0$
$t \leftarrow 0$
while θ_t *not converged* **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$
  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (with $(\cdot)^2$ as the component-wise square)
  $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Bias-correct first moment estimate)
  $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Bias-correct second moment estimate)
  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / \left(\sqrt{\hat{v}_t} + \epsilon\right)$
return θ_t

We then modify the Algorithm 4.6 to consider our objective function $f : \mathcal{S}^n_{X,Y} \to \mathbb{R}$ and use the exponential map defined at the current point $x_t \in \mathcal{S}^n_{X,Y}$ for position updates. The gradient $grad_{x_t}(f)$ may not be part of the tangent space $T_{x_t}\mathcal{S}^n_{X,Y}$ but can be projected onto this space in preparation for passing into the exponential map. As we expect very small movements on the sphere, we can simplify the parallel transport with the projection of the state onto the next tangent space as previously mentioned.

The gradient required in the resulting Algorithm 4.7 will be computed using methods presented in Chapter 5. It is important to notice that compared to the regular ADAM implementation, the element-wise square of the gradient is swapped for an inner product. Leaving it to an element-wise square would lead to accumulating state vectors requiring parallel transport which could result in negative values incompatible with taking the root in the next optimization step. This simplifying change may cause this algorithm to lose some of its adaptivity as the second moment estimate no longer gives values per individual vector entry. However, as we will show in Chapter 6 it is still efficient enough. This change is inspired by the implementation of RADAM [21] and VectorADAM [24].

Algorithm 4.7: Modified ADAM to work in the ambient space of a Riemannian manifold. Recommended values stay the same as in Algorithm 4.6

require : α: Stepsize, $\beta_1, \beta_2 \in [0, 1)$: Decay rates for moment estimates, ϵ: Numerical stability additive
require : $f(x) : \mathcal{S}^n_{X,Y} \to \mathbb{R}$: Objective function on point x, x_0: Initial point vector
require : $\Pi_x(v)$: Projection of v in the ambient space onto $T_x\mathcal{S}^n_{X,Y}$
require : $\|\cdot\|_{\mathcal{S}}$: Norm of the given $\mathcal{S}^n_{X,Y}$

$m_0 \leftarrow 0$
$v_0 \leftarrow 0$
$t \leftarrow 0$
while θ_t *not converged* **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \Pi_{x_{t-1}}(gradf(x_{t-1}))$
  $\tilde{m}_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
  $\tilde{v}_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot \|g_t\|^2$
  $\hat{m}_t \leftarrow \tilde{m}_t / (1 - \beta_1^t)$ (Bias-correct first moment estimate)
  $\hat{v}_t \leftarrow \tilde{v}_t / (1 - \beta_2^t)$ (Bias-correct second moment estimate)
  $\tilde{x}_t \leftarrow \exp_{x_{t-1}} -\alpha \cdot \hat{m}_t / \left(\sqrt{\hat{v}_t} + \epsilon\right)$
  $x_t \leftarrow x_t / \|x_t\|_{\mathcal{S}}$ (normalize to ensure staying on the manifold)
  $m_t \leftarrow \Pi_{x_t}(\tilde{m}_t)$ $v_t \leftarrow \tilde{v}_t$
return x_t

Note that this approach is different from the Riemannian ADAM generalization presented in [21] which aims to implement ADAM on the manifold itself, not concerning the ambient space. We fix the reference frame with our canonical coordinate system of the ambient space of $\mathcal{S}^n_{X,Y}$. Their concerns about moving reference frames on a simple coordinate-wise update thus do not apply. Note that due to these concerns, their method requires additional structure on the manifold in question. In particular, the manifold has to be a cartesian product of the Riemannian manifolds $(\mathcal{M}_i, g^i)$ which is not possible for a hypersphere as in our case.

5 Implementation

In this chapter, we highlight important parts of the implementation and explain how we accelerate computation of both the objective functions from Chapter 3 and their gradients. In particular, we present certain modifications made to a library for sector length computation helpfully provided by Nikolai Wyderka [25]. The implementation used to obtain the results in Chapter 6 can be found on GitHub [26].

5.1 Automatic Differentiation and Compilation for Python

Regardless of the particular implementation of the objective function used, the main challenge is calculating the gradient at any given point. Calculating the gradient in an efficient fashion is central to using gradient based optimization methods. In theory any given function implementation could be manually differentiated, but keeping the function and the derivative implementation synchronized can be cumbersome and time-consuming. A common approach to resolving this is using automatic differentiation frameworks, so called "auto-graders", in order to programmatically calculate the derivative of any given function. Depending on the used framework for auto-grading, several benefits and restrictions apply in comparison to other frameworks.

The implementation of the objective function variants employed within this work is given using the library JAX from Google [27]. This framework is constructed as a project furthering machine learning capabilities by offering automatic differentiation through a large subset of native Python and Numpy [28] code. Additionally, JAX can compile most of the automatically differentiable Python and Numpy code just in time (JIT) using XLA from Tensorflow to kernels fit for execution on hardware accelerators like graphics processing units (GPUs) and tensor processing units (TPUs). This JIT compilation happens automatically when using a specific implementation of Numpy packaged with the JAX library, but it can also be used as a tool to reduce larger, self written functions into single executable kernels.

JAX performs its JIT compilation using a two-step process. First, native Python and Numpy code is *lowered* to XLA primitives. In the special implementation of the Numpy library given by JAX the lowering step may have been performed manually beforehand to ensure an efficient translation for highly used numerical operations. The second stage *compiles* the sequence of XLA primitives which could also be individually compiled and executed on a hardware accelerator in a process called *fusion*. The result is a single operation, enabling it to be launched in a single call to the hardware accelerator. Removing the need for synchronization and additional kernel launches by the orchestrating CPU significantly reduces the overhead produced by an implementation using a high count of singular primitives to be executed on a hardware accelerator. This single operation is then compressed further and internally optimized by removing write-outs of intermediate data structures to memory, instead streaming results directly to their using sinks. Together, both optimizations can speed up native Python code significantly by efficiently using compute and bandwidth resources available on modern hardware accelerators. However, fusion into a single operation also means that at any state in the compiled kernel execution the intermediate data structures produced by the program must fit into memory given

onboard the hardware accelerator. Thus, JAX often relieves pressure when dealing with runtime scaling issues with by shifting it to memory scaling issues as hardware accelerators often come with less on board memory than is given in main memory.

Building and improving the capability for automatic differentiation inside JAX is an ongoing research topic. Currently, differentiation as a JAX function transformation works by wrapping function arguments in *tracer objects* which record any operation they are a part of. When creating intermediate data structures like computing the density matrix from a state vector in preparation for computing its reduced density matrices, tracer objects will multiply and fill in the entire density matrix, leading to an exponentially scaling memory overhead. As will be explained in Chapter 6, this memory overhead will limit the number of qubits n and the order k for which computations of the sector lengths are feasible with current hardware.

5.2 Implementing state optimization

We obtained a Python library from Nikolai Wyderka [25] that can compute sector lengths in terms of purities as explained in Section 3.3. This library works for native Python calls, but certain parts of it needed rewrites in order to be fully compilable with JAX transformations. As an example, the nature of JAX tracers disallows them to be naturally cast to integers or used for array indexing.

The relevant code given by the library is extracted and rewritten to fit the requirements above. Additionally, the optimizations for symmetric states and reducing the number of purities that need computation from Section 3.4 were implemented.

We want to highlight that we store $n+1$ parameters for a symmetric state as mentioned in Section 3.5. From these parameters, the full state vector with 2^n parameters needs to be reconstructed for computation of sector lengths. Since this reconstruction will be executed at every step, we precompute the Dicke state for each of the $n+1$ parameters. Using the obtained library, computing the Dicke state D_k^n necessitates constructing a binary string $\underbrace{0\ldots0}_{n-k}\underbrace{1\ldots1}_{k}$ and finding every permutation of it. These combinations are then turned into the computational basis states corresponding to the binary string, added together and normalized. However, constructing and handling these binary strings is quite expensive. Instead, we construct the integer set $\{2^i | i \in \{0 \leq i < n\}\}$ and receive all its subsets of k elements. For each subset, we sum its elements to receive an index in the general state vector in which an excitation appears in the Dicke state. Using integers instead of strings and avoiding unnecessary state construction enables scaling computation of the Dicke states beyond $n = 10$.

Another important part of the implementation is the exponential map which we use to move on the spheres constructed in Sections 4.2 and 4.3. We can think of the geodesic starting at some anchor x with an initial tangent direction $0 \neq v \in T_x\mathcal{S}^n_{X,Y}$ as a great circle of speed $\|v\|$ in the Euclidean plane that is spanned by x and v. Thus, the exponential map for this geodesic is given by the following closed form:

$$\gamma(t) \coloneqq \cos(\|v\|t)x + \sin(\|v\|t)\frac{v}{\|v\|}.$$

Finally, the implementation of our modified algorithm from Section 4.4 is inspired by the “geoopt” library [29] which implements the RADAM algorithm presented in [21] for the PyTorch [30] machine learning framework. We used the framework “Hydra” [31] for configuration and batch dispatching of our optimization runs.

6 Experiments

In this chapter we will use the sector length computation methods from Chapter 3, the new optimization technique from Chapter 4, and our efficient implementation from Chapter 5 to explore bounds on sector lengths. In particular, we present findings that support Conjecture 3.5. Additionally, we apply different restrictions to the optimization and record their impact on optimization speed, resource usage, and the achieved sector length values. We also evaluate several potential improvements to the optimization. Lastly, we compare our bounds on sector lengths that we found numerically to those produced by shadow inequalities.

6.1 Exploring sector lengths

To recall, the main goal of this thesis is to support the following Conjecture 3.5 by providing lower bounds and suggesting n_0 for different k.

Conjecture (See Conjecture 3.5)**.** *For all k there exists a n_0 such that for all $n \geq n_0$, $A_k \leq \binom{n}{k}$ holds for every state of n qubits.*

Indeed, we found such n_0 using the methods described in Chapters 3 and 4:

Result 6.1. *We present our suggested values of n_0 for different k, supporting the Conjecture 3.5:*

Sector	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9
LB on n_0	*3*	*5*	*8*	*10*	*11*	*13*	*15*	> 16

Even if some of these values do not prove correct and n_0 is even higher or does not exist these values still constitute a lower bound as for every $n' < n_0$ for our suggested n_0, $\binom{n'}{k}$ does not constitute a bound on the sector length A_k. Note that the values for $k = \{2, 3\}$ are already proven correct as their relevant state spaces have been fully characterized [6].

We also give an example for a full optimization table using $k = 4$ for different n in Table 6.2. The optimization runs used to obtain this optimization table are presented in Fig. 6.3. Upper bounds up to $n = 6$ are already known [32], but can be recreated here and matched with a lower bound. The shadow inequalities discussed in Section 3.2 only achieve non-tight upper bounds for $n > 6$ [6]. For the first few $n > 6$ we provide lower bounds on the maximal values of A_4 and suggest these bounds are tight.

For $k = 9$ we managed to investigate maximal values up to $n = 16$ and found that they still exceed $\binom{n}{k}$, thus we conclude that $n_0 > 16$. These investigations were done with the simplifications discussed in Section 6.2. Even with those simplifications however, scaling this to higher n quickly becomes difficult.

n	$\max_{\|\psi\rangle} A_4\|\psi\rangle$	nCk
4	9	1
5	15	5
6	45	15
7	45	35
8	70	70
9	126	126
10	210	210

Table 6.2: Maximal values of A_4 attained using unconstrained optimization methods from Chapter 4 for various n. Note that for $n \geq 8$, $A_4 \leq \binom{n}{4}$ appears to hold, thus we suggest $n_0 = 8$ for $k = 4$.

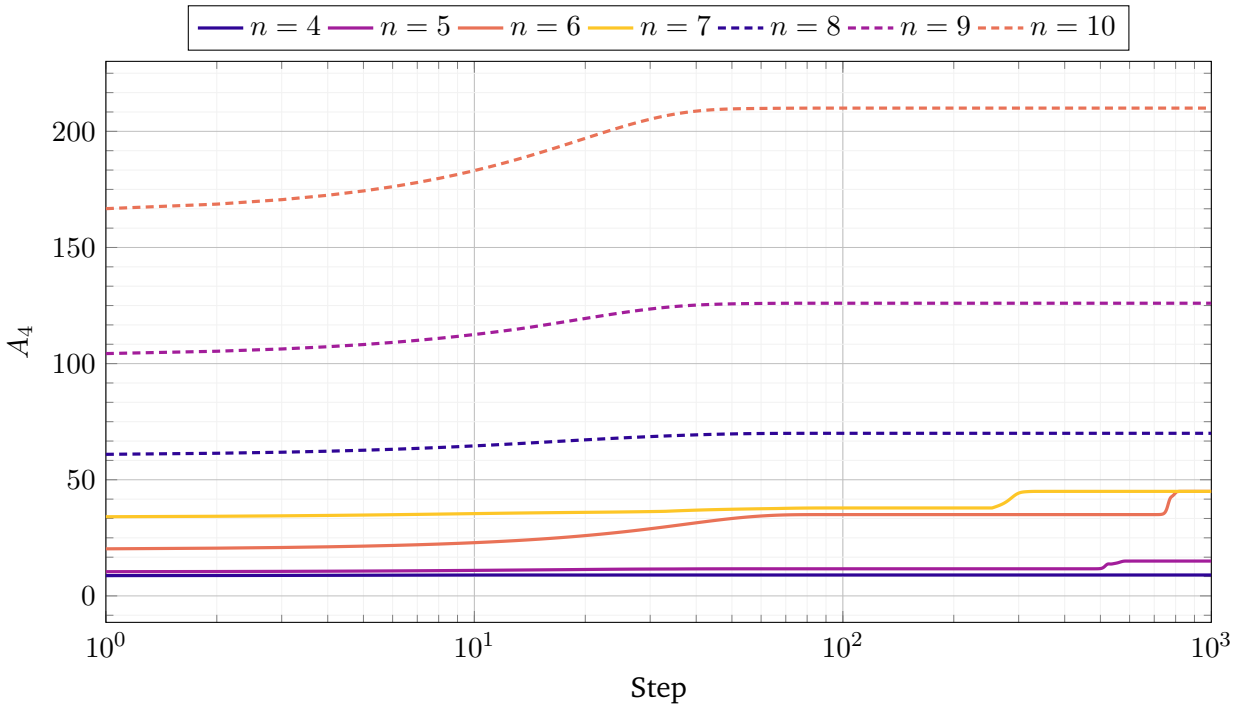

Figure 6.3: Maximal values from optimization runs for A_4. Each run set contains 100 optimization runs.

Due to the number of individual values during calculation increasing much faster than the magnitude of A_k for rising n and k, calculating A_k involves increasingly smaller numbers. This quickly leads to precision issues, where optimization is not effective anymore as the range of possible stepping directions that are indistinguishable with regular 32-bit precision floating point numbers (further referred to as float32) becomes quite large. The only option to combat this is increasing precision by switching to 64-bit precision floating point numbers (further referred to as float64).

We show the impact of switching from float32 to float64 values in Fig. 6.4. We observe a much higher stability and faster convergence using float64 as less stepping directions are indistinguishable. Thus, with float64 values the intended stepping direction is more resistant to numerical error. Note that such great discrepancies usually only occur for $k \geq 7$. For the comparison presented we leveraged the improvement in performance through restriction to symmetric states and real parameters as introduced in Section 6.2 since properly exploring sector A_7 takes a long time without them. However, several test runs showed that the difference in stability is

observable for unrestricted optimization as well.

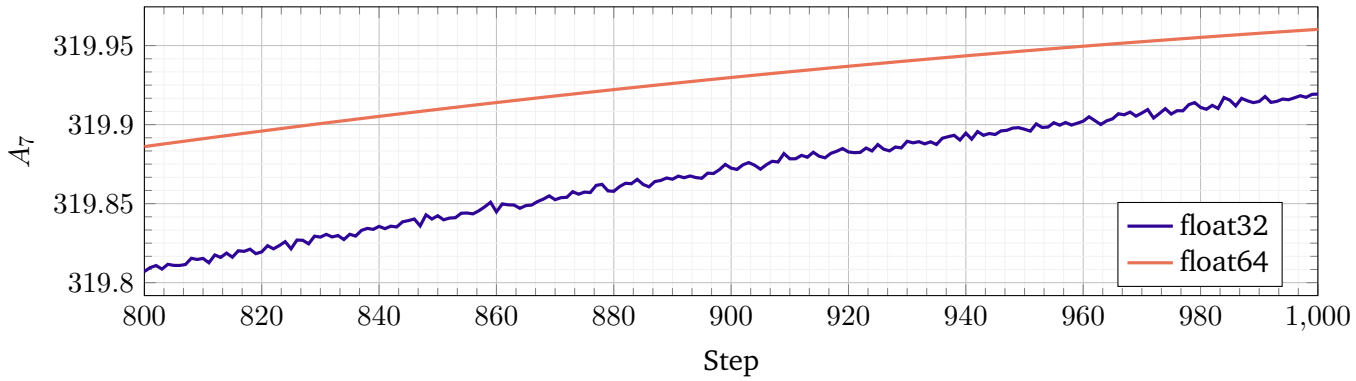

Figure 6.4: Comparison of mean float32 and float64 optimization runs for $n = 10$ and $k = 7$ zoomed in on the last 200 steps. 10 runs were used for each graph.

Note that this change naturally increases the time and memory required for some of the steps in an optimization run. We display the time difference for optimization runs on A_7 in both the lowering and the raw optimization time in Fig. 6.5.

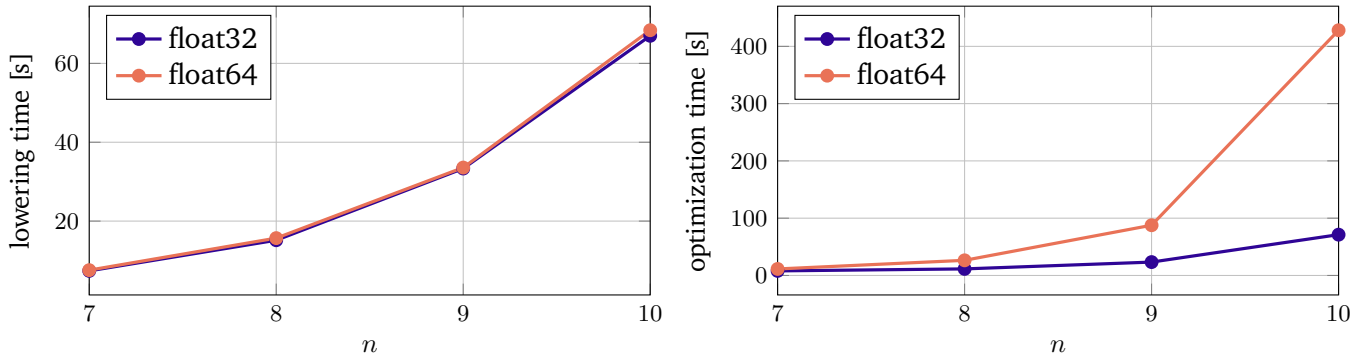

Figure 6.5: Comparison of mean time required in optimization runs in A_7 for float32 and float64 respectively. Note that the lowering time barely increases, while the optimization time is heavily impacted by the elevated precision. Values are obtained using 10 runs for each graph.

We also record the peak memory footprint in Fig. 6.6 which naturally increases with double precision. The JAX library itself consumes about 238 Mebibyte (MiB) of hardware accelerator memory and a small quantity of CPU memory. We do not list the required CPU memory as the requirements for hardware accelerator memory scale much faster. When increasing n one will rather run out of hardware accelerator memory than CPU memory. If not specified otherwise, every result presented in this thesis has been obtained using 64-bit precision.

6.2 Exploring optimizations

Even without switching to float64, the time required to do a full optimization run including lowering, compilation, and differentiation of the sector length objective function quickly becomes infeasible with rising n and k. The time allocated for the preparatory steps before the optimization itself quickly amounts to a large

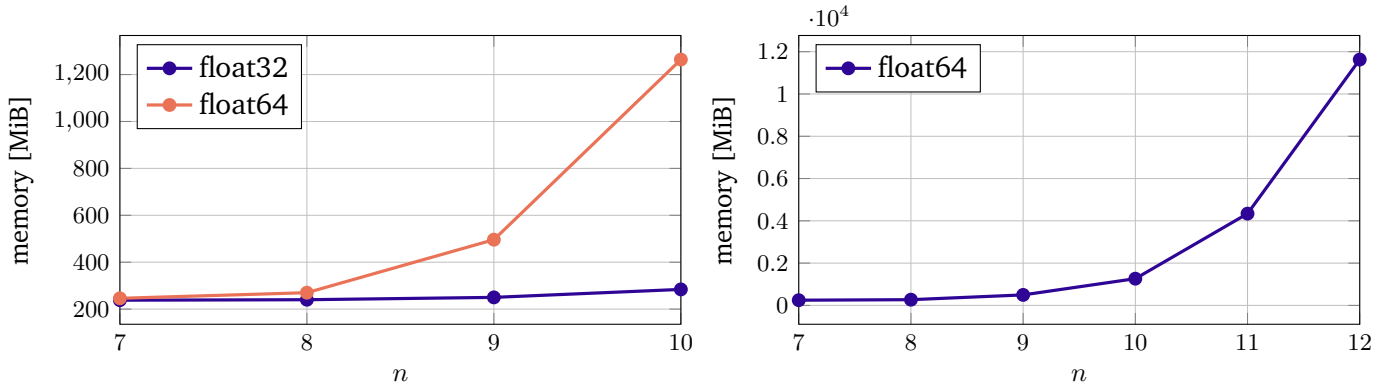

Figure 6.6: Left: comparison of peak memory required in optimization runs in A_7 for float32 and float64 respectively. Note that for the JAX library, a base value of 238 MiB is required. Right: peak memory required in optimization runs in A_7 with float64 for higher n.

part of the total runtime. Thus, reducing the time spent with such preparatory steps is paramount to speeding up overall computations across many runs.

As discussed in Chapter 3, computation of sector lengths may be optimized in various fashions. In particular, we can compute sector lengths over purities of system reductions instead of using expectation values with full n-Pauli operators (referring to the Ξ_k which are expensive to compute and store). However, some possibilities for optimization have the benefit of reducing the size of the state space searched during optimization using the techniques discussed in Chapter 4. Good instances of this are the reduction of state space by restricting the optimization to only use real parameters (using $\mathcal{S}^n_{\cdot,\mathbb{R}}$) or symmetric states (using $\mathcal{S}^n_{S,\cdot}$). Naturally, the question arises whether sector lengths reachable without any restrictions are still reachable with restrictions in place. As an example, restricting the search to pure states does not influence the range of attainable sector length values [2].

To investigate the impact of further restricting the searchable state space, we first compare optimized values specifically for the sector length A_4, as the obtainable state space structures for the first two sector lengths of interest A_2 and A_3 are fully characterized already [6]. Obtained maximal values for sector length A_4 for different n and with different restricted state sets are displayed in Table 6.8. For all n but $n \in 6, 7$, obtained values for all different configurations coincide. For the remaining $n \in 6, 7$, we obtain the following result:

Result 6.7. *Differing maximal values for unrestricted, pure state space and restricted, symmetric states space (see Table 6.8) already show that searching symmetric state space may be insufficient for obtaining the highest possible sector length in every (n, k) configuration.*

Similar to Table 6.8, Table 6.9 presents obtained maximal values of A_7 for different n. Note that although there is no visible impact on the attainable maxima of restricting optimizations to real parameters, we find in our experiments that restrictions in this manner most often result in subpar performance in terms of speed of convergence when compared to unrestricted optimizations. This effect is usually more noticeable when not restricting optimizations to symmetric states. A possible reason for this behaviour could be the additional degrees of freedom that are given with complex parameters in comparison to real parameters.

For $k = 7$, only $n \leq 10$ are listed here, as scaling computations past this threshold for less restricted (in particular not only restricted to symmetric states) optimizations quickly becomes infeasible. This is mostly due to the lowering and compilation steps for the objective function requiring exponentially more time with rising

n	$\lvert\psi\rangle \in \mathcal{S}^n_{P,\mathbb{C}}$	$\lvert\psi\rangle \in \mathcal{S}^n_{P,\mathbb{R}}$	$\lvert\psi\rangle \in \mathcal{S}^n_{S,\mathbb{C}}$	$\lvert\psi\rangle \in \mathcal{S}^n_{S,\mathbb{R}}$	nCk
4	9	9	9	9	1
5	15	15	15	15	5
6	45	45	35	35	15
7	45	45	37.857	37.857	35
8	70	70	70	70	70
9	126	126	126	126	126

Table 6.8: Numerical $\max_{|\psi\rangle} A_4(|\psi\rangle)$ for different $n \leq 9$ and $|\psi\rangle \in \mathcal{S}$ for $\mathcal{S} \in \{\mathcal{S}^n_{P,\mathbb{C}}, \mathcal{S}^n_{P,\mathbb{R}}, \mathcal{S}^n_{S,\mathbb{C}}, \mathcal{S}^n_{S,\mathbb{R}}\}$. Values were obtained using 20 optimization runs in each configuration. For comparison, we also display nCk here. Note that all values conincide at nCk for $n \geq n_0$ using our suggested $n_0 = 8$ for A_4.

n	$\lvert\psi\rangle \in \mathcal{S}^n_{P,\mathbb{C}}$	$\lvert\psi\rangle \in \mathcal{S}^n_{P,\mathbb{R}}$	$\lvert\psi\rangle \in \mathcal{S}^n_{S,\mathbb{C}}$	$\lvert\psi\rangle \in \mathcal{S}^n_{S,\mathbb{R}}$
7	64	64	64	64
8	84.571	84.571	84.571	84.571
9	182	182	137.111	137.111
10	360	360	320	320

Table 6.9: Numerical $\max_{|\psi\rangle} A_7(|\psi\rangle)$ for different $n \leq 10$ and $|\psi\rangle \in \mathcal{S}$ for $\mathcal{S} \in \{\mathcal{S}^n_{P,\mathbb{C}}, \mathcal{S}^n_{P,\mathbb{R}}, \mathcal{S}^n_{S,\mathbb{C}}, \mathcal{S}^n_{S,\mathbb{R}}\}$. Values were obtained using 20 optimization runs in each configuration.

n independent of k. As a simple improvement in a multi-run scenario one might add JAX-native compilation caching with which JAX will recognize equivalence between already compiled functions and the function to be compiled after the lowering step. This still requires the function to be fully lowered every run and fully compiled at least once. While we used a compilation cache for most optimization runs, we disabled it for time and memory measurements.

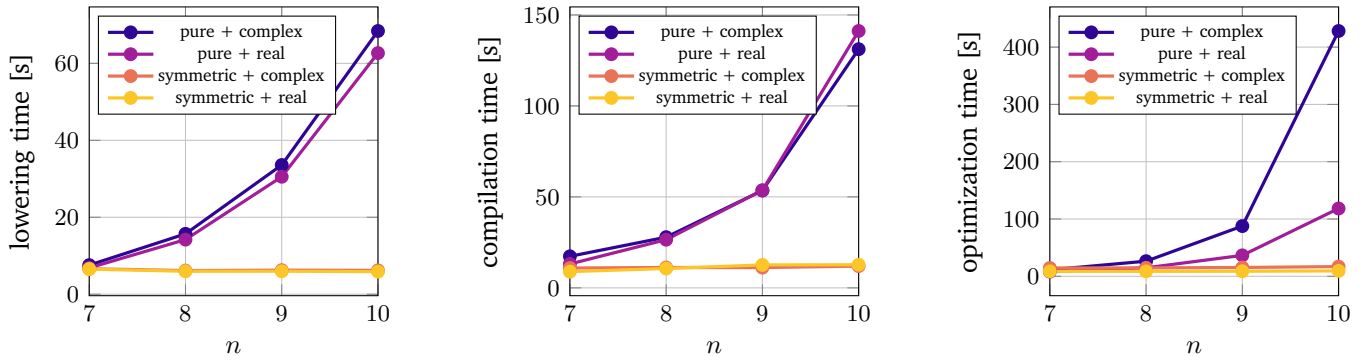

Figure 6.10: Comparison of mean time required in optimization runs in A_7 for different restrictions. We intentionally leave out grading time as it is implied by individually lowering and grading the objective function. Values are obtained using 10 runs for each graph.

However, the impact of restricting optimization is much more noticeable. Restricting optimizations to symmetric states has a significant impact on all phases of a full optimization run due to requiring only one marginal to be computed (see Theorem 3.12). We record this impact in Fig. 6.10. Note that we leave out grading time, as we lower and compile the sector length objective function individually which eliminates grading time. When the steps are computed individually, both lowering and compiling the function have to be done with a given data sample and the steps automatically grade the function. Restricting optimizations to real parameters does not

have such a significant impact on lowering and compilation as the difference in computational complexity is minimal. However, restriction to real parameters still incurs a noticeable speedup for pure optimization time.

With the restriction to symmetric states, our limiting factor is not the time available for compilation, but the memory available to our hardware accelerators. The restrictions do not eliminate the memory scaling problem but merely delay it. We display the peak memory required for optimization runs on A_7 for select n in Fig. 6.11 on the left side. Note that for restriction to symmetric states, the difference in required memory is so significant that it enables us to scale computations to a few higher n than previously. We record the memory required for higher n and A_7 on the right side of Fig. 6.11.

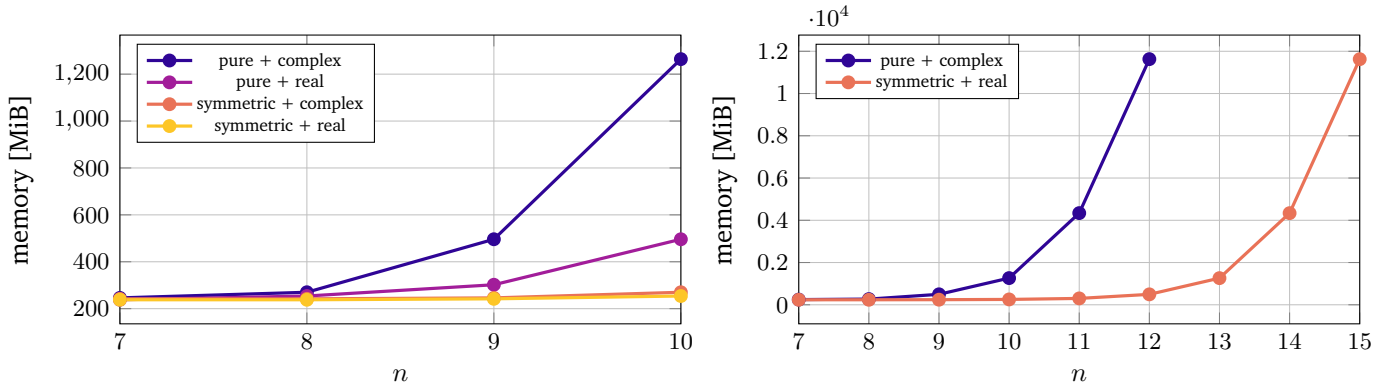

Figure 6.11: Left: comparison of peak memory required in optimization runs in A_7 for different restrictions respectively. Note that for the JAX library, a base value of 238 MiB is required. Right: peak memory required in optimization runs in A_7 with restriction to symmetric states.

We can formulate the following observation based on the values shown for $k = 4, 7$ in Tables 6.8 and 6.9 and our remaining sufficiently investigated $k \leq 8$:

Observation 6.12. *Consider a specific configuration of n, k. If using unrestricted optimization we find a maximal value exceeding $\binom{n}{k}$, we also find the maximal value exceeding $\binom{n}{k}$ using all three restriction combinations.*

Additionally, we notice the following:

Remark 6.13. *Naturally, if $\max_{|\psi\rangle} A_k(|\psi\rangle)) > \binom{n}{k}$ for a fixed n holds in restricted optimization, it must also hold in unrestricted optimization, as restricted state space is by definition fully contained in unrestricted state space.*

Observation 6.12 and Remark 6.13 lead us to the following conjecture:

Conjecture 6.14. *The maximal value of A_k attainable for a given n, k using states in $\mathcal{S}^n_{P,\mathbb{C}}$ exceeds $\binom{n}{k}$ if and only if the maximal value attainable using states in the restricted space $X \in \{\mathcal{S}^n_{P,\mathbb{R}}, \mathcal{S}^n_{S,\mathbb{C}}, \mathcal{S}^n_{S,\mathbb{R}}\}$ also exceeds $\binom{n}{k}$. More formally, this can be expressed as*

$$\forall n, k \text{ with } n > k:$$
$$\max_{|\psi\rangle \in \mathcal{S}^n_{P,\mathbb{C}}} A_k(|\psi\rangle) > \binom{n}{k} \Leftrightarrow \max_{|\psi\rangle \in \mathcal{S}^n_{P,\mathbb{R}}} A_k(|\psi\rangle) > \binom{n}{k} \Leftrightarrow \max_{|\psi\rangle \in \mathcal{S}^n_{S,\mathbb{C}}} A_k(|\psi\rangle) > \binom{n}{k} \Leftrightarrow \max_{|\psi\rangle \in \mathcal{S}^n_{S,\mathbb{R}}} A_k(|\psi\rangle) > \binom{n}{k}.$$

Originally, just by using Remark 6.13, we are able to use the speedup from optimization with state space restriction to our benefit. We can quickly raise the lower bound on n_0 as we eliminate candidates for it. If Conjecture 6.14 proves true, we can also soundly investigate surrounding n for some candidate for n_0 without leaving restricted state space. This in itself would be a great improvement regarding memory consumption and runtime, especially in combination with optimizations for symmetric states proposed at the end of Section 3.4. Note that this conjecture is in particular not in conflict with Result 6.7 as we still do not require that for *every* configuration of (n, k), using restricted optimizations has no impact on the attainable maxima.

6.3 Kickstarting state optimization

During optimization runs for the specific configuration $k = 5, n = 9$, we found a discrepancy between the possible performance of optimizations with and without restriction to symmetric state space. Symmetric state space optimizations consistently achieved values for $A_5 > 126$, though by a small amount, whereas unrestricted optimization was consistently bound by $A_5 \leq 126$ as can be seen in Fig. 6.15. As such results are unsound we have to search for methods that boost unrestricted optimization sufficiently.

One such method we found is *kickstarting* the optimization by searching through unrestricted state space, but starting with a random symmetric state instead of a random unrestricted state. In the scenario $k = 5, n = 9$, the maximal value is likely to be attained by a symmetric state. Starting with a symmetric state in such cases supports the optimization in reaching these maximal states. This change in start values allowed unrestricted optimization to reach values of $A_5 > 126$ as well, which soundly proved $n_0 > 9$ for $k = 5$ as $\binom{9}{5} = 126$. In Fig. 6.15 we display this impact of kickstarting as well. Note that the resulting states after such optimizations are not necessarily symmetric themselves.

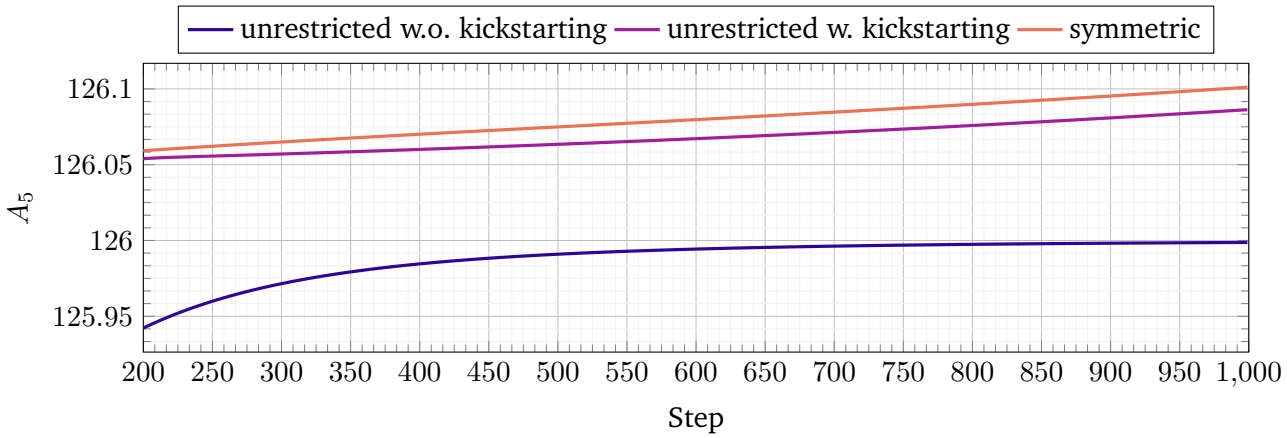

Figure 6.15: Comparison of mean optimization runs for A_5 with $n = 9$ zoomed in on the last 800 steps. 10 runs were used for each graph.

This method of kickstarting unrestricted optimization has no apparent effect on most configurations. However, in scenarios where the maximal value is not attained by a symmetric state, kickstarting can have a negative impact. Take as an example the configuration $k = 4, n = 7$, where kickstarting the unrestricted optimization limited achieved values to $A_4 \leq 37.857$, exactly the maximal value attained for symmetrically restricted optimizations (see Table 6.8). Not using kickstarting returned the unrestricted optimizations to its true maximally attained value of $A_4 \leq 45$. This impact on $k = 4, n = 7$ is validated using 100 optimization runs both for enabling and disabling kickstarting. We display the maximal values attained in those runs in Fig. 6.16.

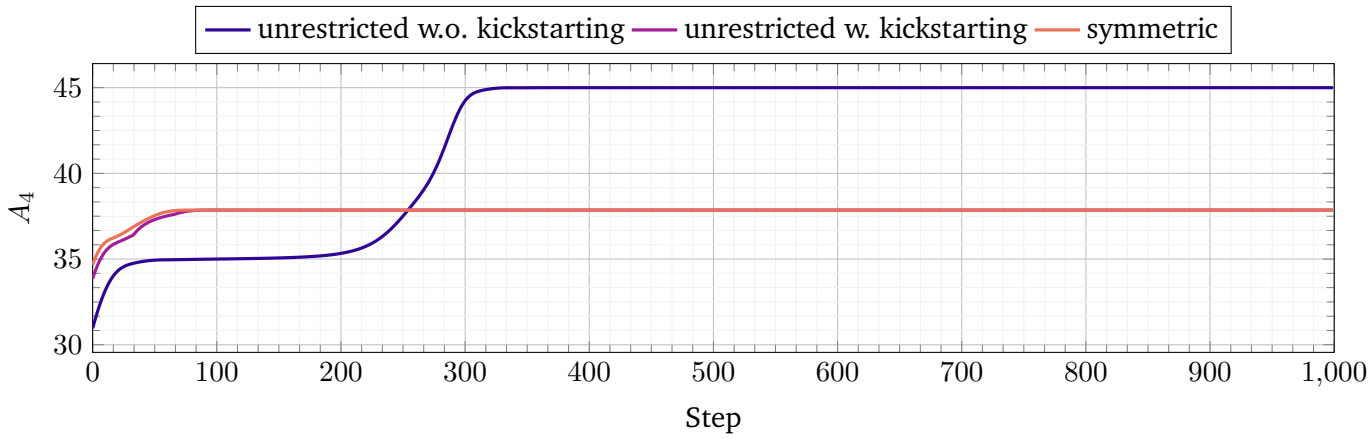

Figure 6.16: Comparison of different optimization runs for A_4 with $n = 7$. 100 runs were used for each graph.

Kickstarting is also possible between different restrictions of state space, as long as the state space used to provide the initial state is fully contained in the state space in which to optimize. Kickstarting might have benefits when used with the restriction of state space to real state parameters in comparison to its unrestricted dual with complex state parameters. This needs to be validated further once a configuration with a similarly unsound discrepancy to $k = 5, n = 9$ between real and complex parameters is found.

To summarize, we can use kickstarting when observing an unsound discrepancy between less and more restricted optimizations, i.e. when the more restricted optimization achieved better values for a sector length than its less restricted counterpart. However, kickstarting can also have a negative impact on the achievable values for this less restricted optimization. Thus, we recommend that kickstarting is only used to investigate unsound values.

6.4 Comparison to shadow inequalities

As discussed in Section 3.2, shadow inequalities are a useful tool for finding upper bounds to correlations that can be found in a given state. These shadow inequalities can be used to find n_0 for $k \in \{2, 3\}$. For higher k, they can be obtained but seem to fail to provide any upper bound for n_0 when k is even. For sector length A_4, it is proven that shadow inequalities are insufficient to show our suggested $n_0 = 8$ [6].

For odd $k \leq 13$, we were able to leverage shadow inequalities to find upper bounds on n_0. However, when investigating the upper bounds produced by shadow inequalities, it becomes evident that our suggested values for n_0 are significantly lower than approximated by the current upper bounds. As an example, we obtained our suggested $n_0 = 10$ for A_5, but shadow inequalities only manage to lower the upper bound to $n_0 \leq 12$. This is especially true for higher k, as presented in Table 6.17.

6.5 Additional experiments

In addition to the impact on optimization of different state space restrictions investigated so far, one can alter the algorithms used for optimization. In particular, instead of fully implementing the modified ADAM from

k	LB on n_0	UB on n_0
3	5	5
5	10	12
7	13	20
9	>16	29

Table 6.17: Lower bounds on n_0 for different odd k where we suggest these values as valid n_0. Additionally, we display upper bounds on n_0 obtained from shadow inequalities.

Algorithm 4.7, requiring implementation of parallel transport on the manifold, the simpler SGD algorithm in its Riemannian implementation [20] could be used. However, the performance of Riemannian stochastic gradient descent (RSGD) seems subpar compared to our modified ADAM according to the earliest of our experiments. Whilst RSGD may come close to our found maximal values faster than ADAM, it lacks the stability and long-term proximity to the target value that ADAM provides.

Regardless of the concrete optimization method used, one can alter the step size (often called *learning rate* in the context of optimization methods) of movement on the manifolds. Specifically, the step size could be altered with each iteration, implementing e.g. a decaying schedule. This modification is also known as *learning rate decay*. It could improve performance of the optimization methods since it often improves results in machine learning applications [33]. In our experiments, we observed no noticeable improvement from using e.g. an exponential decay schedule. Choosing the step size of 10^{-2} which is recommended for most ADAM applications seems to give good results for most of the configurations we investigated.

These additional results and suggestions need to be investigated further and validated independently.

7 Future extensions

The most promising proposed improvement is that of removing the dependency of sector length computation of symmetric states on calculation of reduced density matrices (see Section 3.5). This would increase the scalability of sector length explorations and elimination of n_0 candidates (if Conjecture 6.14 proves true, also suggestion of n_0) by a large margin.

However, we can also improve our computation by other means. As an example, we can try to cache as many parts of the computation as possible after computing the objective function and its derivative once, reducing the time overhead besides the optimization steps. This may be realized with a method in Python development called *pickling*, in which an object can be serialized to a byte stream, allowing it to be saved to an external file. Additionally, computation of the objective function and its derivative may be moved to machines with higher single-thread performance while optimization remains on machines with higher hardware accelerator and parallelization performance. This would also require migrating from JIT to ahead of time (AOT) compilation. Parallelization of lowering and compilation may be possible with the recent movement of removing the Global Interpreter Lock of Python which previously prevented any parallel interpretation of python code and thus JAX compilation. Note that JAX support for such means and operations is still sparse.

As mentioned in Section 6.3, kickstarting optimizations using complex parameters with states only using real parameters may become useful. However, unsound discrepancies between real and complex parameters are first needed to observe any improvements. Alternatively, for improvements to the optimization methods different step sizes on the manifolds as well as non-constant step size scheduling may prove useful and need to be investigated.

With all computations covered in this thesis, objective functions were formed from expressions yielding a single sector length for some n, k in *qubits*. Generalising such computations and the concept of sector lengths to *qudits* is possible, thus the methods presented here deserve to be generalised as well. For example, the spheres used to represent usable states with or without some restrictions themselves can easily be generalised to take an additional parameter d. This also motivates search in spaces which are further or differently restricted, such as *n-separable* state space. Some numerical investigations of bounds on sector lengths with separable states have already been made [32]. However, these investigations were mostly done through informed guesses, thus limiting their power compared to automated optimization. Implementing additional restrictions on state space may prove useful and needs investigation as such restrictions may allow speeding up optimizations even further.

Finally, optimizations can be generalised from a single sector length value to arbitrary expressions of such values by simply combining multiple individual objective functions into a single, larger objective function. Such expressions could be entropies or entanglement measures [6], enabling raising suprema on those measures and further understanding their complex behaviour and structure.

8 Conclusions

We introduced new optimization methods for arbitrary pure quantum states given real-valued objective functions in Chapter 4 and implemented those objective functions for individual sector lengths in Chapters 3 and 5. Additionally, we discussed possible optimizations to the objective functions that significantly reduced the amount of computational resources needed for them. With the new optimization methods, we numerically investigated maximally attainable values of individual sector lengths for different configurations of (n, k) in Section 6.1. Using these maximally attainable values, for $4 \leq k \leq 8$ we supported the existence of a threshold $n_0 \leq n$ at and after which $A_k \leq \binom{n}{k}$ holds for states of n parties by suggesting values for n_0 in Result 6.1. For $k = 9$, we provided a lower bound on this threshold n_0. Furthermore, we studied the impact of restricting optimizations on the attainable maximal values in Section 6.2. Using these results, we proposed an equivalence of certain restrictions with unrestricted optimizations for the purposes of finding n_0 for a given k. We showed that restricting the optimization significantly improves the scalability of our approach both in terms of memory usage and overall time required. Lastly, we compared the results of our approach to upper bounds on n_0 produced by shadow inequalities in Section 6.4 and investigated different modifications to this approach in Sections 6.3 and 6.5.

This thesis gives many points of extension for future work, including deeper investigation of different similar optimization methods and the impact of their hyperparameters. We also propose extension of the methods discussed in this thesis to expressions with multiple sector lengths, yielding methods for optimization of entire entanglement and entropy relations.

Algorithms, Figures and Tables

List of Algorithms

List of Figures

List of Tables

Abbreviations

Abbreviations

Notation	Description	Page List
AME	Quantum state such great entanglement, that all reduced systems of at most half the particles are maximally mixed.	1
AOT	Compilation of source code separate from its execution; planned compiliation.	29
GPU	Hardware accelerator usually used for graphical processing. Efficient at highly parallel computations.	18
JIT	Compilation of source code right before its execution; on demand compilation.	18, 29
MiB	2^{20} bytes. Note the difference to the Megabyte, which is 1000^2 bytes.	22, 23, 25
RSGD	Stochastic gradient descent algorithm class adapted to be used with Riemannian manifolds.	28
SDP	Semi Definite Program	12
SGD	Gradient based optimization algorithm class often used in machine learning applications.	12, 28
TPU	Hardware accelerator usually used for tensor applications. Developed by Google specifically for Tensorflow. See https://cloud.google.com/tpu.	18

Bibliography

[1] Valerie Coffman, Joydip Kundu, and William K. Wootters. "Distributed entanglement." In: *Physical Review A* 61.5 (Apr. 2000). URL: https://doi.org/10.1103%2Fphysreva.61.052306.

[2] Hans Aschauer et al. *Local invariants for multi-partite entangled states allowing for a simple entanglement criterion*. 2004.

[3] Felix Huber, Otfried Gühne, and Jens Siewert. "Absolutely Maximally Entangled States of Seven Qubits Do Not Exist." In: *Phys. Rev. Lett.* 118 (20 May 2017), p. 200502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.118.200502.

[4] C. Klöckl and M. Huber. "Characterizing multipartite entanglement without shared reference frames." In: *Phys. Rev. A* 91 (4 Apr. 2015), p. 042339. URL: https://link.aps.org/doi/10.1103/PhysRevA.91.042339.

[5] R. M. Gingrich. "Properties of entanglement monotones for three-qubit pure states." In: *Phys. Rev. A* 65 (5 Apr. 2002), p. 052302. URL: https://link.aps.org/doi/10.1103/PhysRevA.65.052302.

[6] N Wyderka and O Gühne. "Characterizing quantum states via sector lengths." In: *Journal of Physics A: Mathematical and Theoretical* 53.34 (July 2020), p. 345302. URL: https://doi.org/10.1088%2F1751-8121%2Fab7f0a.

[7] Christopher Eltschka et al. "Exponentially many entanglement and correlation constraints for multipartite quantum states." In: *Phys. Rev. A* 98 (5 Nov. 2018), p. 052317. URL: https://link.aps.org/doi/10.1103/PhysRevA.98.052317.

[8] Felix Huber et al. "Bounds on absolutely maximally entangled states from shadow inequalities, and the quantum MacWilliams identity." In: *Journal of Physics A: Mathematical and Theoretical* 51.17 (Mar. 2018), p. 175301. URL: https://doi.org/10.1088%2F1751-8121%2Faaade5.

[9] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017.

[10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[11] Eric M. Rains. *Polynomial invariants of quantum codes*. 1997.

[12] A.R. Calderbank et al. "Quantum error correction via codes over GF(4)." In: *IEEE Transactions on Information Theory* 44.4 (1998), pp. 1369–1387.

[13] E.M. Rains. "Quantum shadow enumerators." In: *IEEE Transactions on Information Theory* 45.7 (1999), pp. 2361–2366.

[14] Felix Huber and Simone Severini. "Some Ulam's reconstruction problems for quantum states." In: *Journal of Physics A: Mathematical and Theoretical* 51.43 (Sept. 2018), p. 435301.

[15] F.J. MacWilliams, F.J. MacWilliams, and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. Elsevier Science, 1978.

[16] Eric M. Rains. *Quantum Weight Enumerators*. 1996.

[17] Peter Shor and Raymond Laflamme. *Quantum MacWilliams Identities*. 1996.

[18] R. H. Dicke. "Coherence in Spontaneous Radiation Processes." In: *Phys. Rev.* 93 (1 Jan. 1954), pp. 99–110. URL: https://link.aps.org/doi/10.1103/PhysRev.93.99.

[19] Paul Skrzypczyk and Daniel Cavalcanti. *Semidefinite Programming in Quantum Information Science*. IOP Publishing, Mar. 2023. URL: https://doi.org/10.1088%2F978-0-7503-3343-6.

[20] Silvere Bonnabel. "Stochastic Gradient Descent on Riemannian Manifolds." In: *IEEE Transactions on Automatic Control* 58.9 (Sept. 2013), pp. 2217–2229. URL: https://doi.org/10.1109%2Ftac.2013.2254619.

[21] Gary Bécigneul and Octavian-Eugen Ganea. *Riemannian Adaptive Optimization Methods*. 2019.

[22] M. Spivak. *A Comprehensive Introduction to Differential Geometry*. A Comprehensive Introduction to Differential Geometry Bd. 4. Publish or Perish, Incorporated, 1975. URL: https://books.google.de/books?id=clbvAAAAMAAJ.

[23] Joal W. Robbin and Dietman A. Salamon. *Introduction to differential geometry*. Oct. 2022.

[24] Selena Ling, Nicholas Sharp, and Alec Jacobson. *VectorAdam for Rotation Equivariant Geometry Optimization*. 2022.

[25] Nikolai Wyderka. "Code provided by Nikolai Wyderka for calculation of sector lengths with purities."

[26] Maximilian Rüsch. *Implementation for the methods described in this thesis*. GitHub. Contains the JAX and original version of the library provided by Nikolai Wyderka. 2023. URL: https://github.com/maximilianruesch/sector-lengths/releases/tag/thesis-v1.

[27] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: http://github.com/google/jax.

[28] Charles R. Harris et al. "Array programming with NumPy." In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. URL: https://doi.org/10.1038/s41586-020-2649-2.

[29] Max Kochurov, Rasul Karimov, and Serge Kozlukov. *Geoopt: Riemannian Optimization in PyTorch*. 2020.

[30] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019.

[31] Omry Yadan. *Hydra - A framework for elegantly configuring complex applications*. GitHub. 2019. URL: https://github.com/facebookresearch/hydra.

[32] Nikolai Wyderka. "Table provided by Nikolai Wyderka containing several approximations of bounds found on sector lengths." Most values were found using educated guesses instead of automated optimization.

[33] Yimin Ding. "The Impact of Learning Rate Decay and Periodical Learning Rate Restart on Artificial Neural Network." In: *Proceedings of the 2021 2nd International Conference on Artificial Intelligence in Electronics Engineering*. AIEE '21. Phuket, Thailand: Association for Computing Machinery, 2021, pp. 6–14. URL: https://doi.org/10.1145/3460268.3460270.