Quantum Machine Learning For Classical Data

Leonard P. Wossnig

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

April 25, 2021

I, Leonard P. Wossnig, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

In this dissertation, we study the intersection of quantum computing and supervised machine learning algorithms, which means that we investigate quantum algorithms for supervised machine learning that operate on classical data. This area of research falls under the umbrella of *quantum machine learning*, a research area of computer science which has recently received wide attention. In particular, we investigate to what extent quantum computers can be used to accelerate supervised machine learning algorithms. The aim of this is to develop a clear understanding of the promises and limitations of the current state-of-the-art of quantum algorithms for supervised machine learning, but also to define directions for future research in this exciting field. We start by looking at supervised quantum machine learning (QML) algorithms through the lens of statistical learning theory. In this framework, we derive novel bounds on the computational complexities of a large set of supervised QML algorithms under the requirement of optimal learning rates. Next, we give a new bound for Hamiltonian simulation of dense Hamiltonians, a major subroutine of most known supervised QML algorithms, and then derive a classical algorithm with nearly the same complexity. We then draw the parallels to recent 'quantum-inspired' results, and will explain the implications of these results for quantum machine learning applications. Looking for areas which might bear larger advantages for QML algorithms, we finally propose a novel algorithm for Quantum Boltzmann machines, and argue that quantum algorithms for quantum data are one of the most promising applications for QML with potentially exponential advantage over classical approaches.

Acknowledgements

Portions of the work that are included in this thesis were completed while I was visiting the Institut Henri Poincaré of the Sorbonne University in Paris.

I particularly want to thank Riam Kim-McLeod for the support and help with the editing of the thesis.

I finally want to thank my family for the continued love and support.

Impact Statement

Quantum machine learning bears promises for many areas, ranging from the health-care to the financial industry. In today's world, where data is available in abundance, only novel algorithms and approaches are enabling us to make reliable predictions that can enhance our life, productivity, or wealth. While Moor's law is coming to an end, novel computational paradigms are sought after to enable a further growth of processing power. Quantum computing has become one of the prominent candidates, and is maturing rapidly. The here presented PhD thesis develops and studies this novel computational paradigm in light of existing classical solutions and thereby develops a path towards quantum algorithms that can outperform classical approaches.

Contents

List of Figures

Chapter 1

Introduction and Overview

In the last twenty years, due to increased computational power and the availability of vast amounts of data, *machine learning* (ML) has seen an immense success, with applications ranging from computer vision [1] to playing complex games such as the Atari series [2] or the traditional game of Go [3]. However, over the past few years, challenges have surfaced that threaten the end of this revolution. The main two challenges are the increasingly overwhelming size of the available data sets, and the end of Moore's law [4]. While novel developments in hardware architectures, such as graphics processing units (GPUs) or tensor processing units (TPUs), enable orders of magnitude improved performance compared to central processing units (CPUs), they cannot significantly improve the performance any longer, as they also reach their physical limitations. They are therefore not offering a structural solution to the challenges posed and new solutions are required.

On the other hand, a new technology is slowly reaching maturity. Quantum computing, a form of computation that makes use of quantum-mechanical phenomena such as superposition and entanglement, has been predicted to overcome these limitations on classical hardware. Quantum algorithms, i.e., algorithms that can be executed on a quantum computer, have been investigated since the 1980s, and have recently received increasing interest all around the world.

One area that has received particular attention is quantum machine learning see e.g. [5], the combination of quantum mechanics and machine learning. Quantum machine learning can generally be divided into four distinct areas of research:
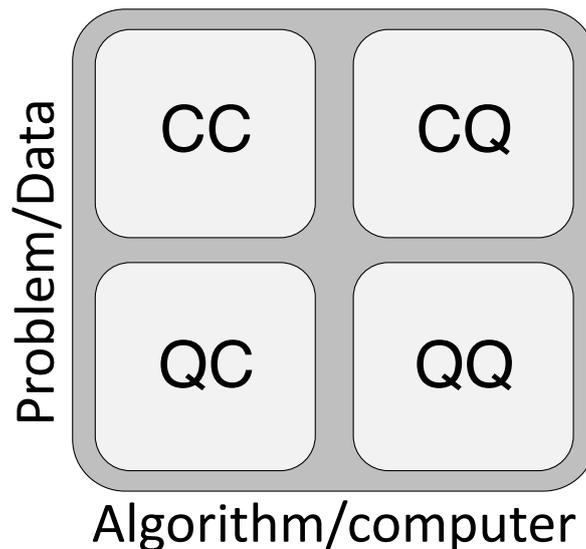
Figure 1.1: Different fields of study in Quantum Machine Learning. The different areas are related to the choice of algorithm, i.e., whether it is executed on a quantum or classical computer, and the choice of the target problem, i.e., whether it operates on quantum or classical data.

- Machine learning algorithms that are executed on classical computers (CPUs, GPUs, TPUs) and applied to classical data, the sector CC in Fig. 1.1,

- Machine learning algorithms that are executed on quantum computers (QPUs) and applied to classical data, the sector QC in Fig. 1.1,

- Machine learning algorithms that are executed on classical computers and applied to quantum data, the sector CQ in Fig. 1.1, and

- Machine learning algorithms that are executed on quantum computers (QPUs) and applied to quantum data, the sector QQ in Fig. 1.1.

The biggest attention has been paid to quantum algorithms that perform supervised machine learning on classical data. The main objective in designing such quantum algorithms is to achieve a computational advantage over any classical algorithm for the same task. This area is the main focus of this thesis, and we will for brevity throughout this thesis refer to this area when we speak about QML.

Although becoming an increasingly popular area of research, supervised QML has in recent years faced two major challenges.

Firstly, initial research was aimed at designing supervised ML algorithms that have a superior performance, where performance was measured solely in terms of the computational complexity (i.e., theoretical speed) of the algorithm with respect to the input size (the problem dimension) and the desired error (or accuracy). When we are speaking about the error in this context, then we generally refer to the distance (for example in norm) of the solution that we obtain through the algorithm to the true solution. For example, for most algorithms we obtain guarantees that if we run it in time $\varepsilon^{-\alpha}$, then the solution will be ε-close. Assuming that the solution is a vector x, and our algorithm produces the vector $\tilde{x}$, this would then imply that $\|\tilde{x} - x\| \leq \varepsilon$. Note that different algorithms measure this error differently, but in general we use the spectral or Frobenius norm as measure for the error throughout this thesis.

However, a challenge to this view is that it is today common wisdom in the classical ML community that faster computation is not necessarily the solution to most practical problems in machine learning. Indeed, more data and the ability to extrapolate to unseen data is the most essential component in order to achieve good performance. Mathematically, the behaviour of algorithms to extrapolate to unseen data and other statistical properties have been formalised in the field of statistical learning theory. The technical term of this ability is the so-called generalisation error of an algorithm, and it is possible to obtain bounds on this error for many common machine learning algorithms. The first research question of this thesis is therefore the following:

Research Question 1 (QML under the lens of SLT)**.** *Under the common assumptions of statistical learning theory, what is the performance of supervised quantum machine learning algorithms?*

Most quantum algorithms are described in the so-called query model. Here, the overall computational complexity, i.e., the number of steps an algorithm requires to completion, is given in terms of a number of calls (uses) of a different algorithm, which is called the oracle. Indeed, most known quantum machine learning algorithms heavily rely on such oracles in order to achieve a 'quantum advantage'. A

second challenge is therefore posed by the question how much of the quantum advantage stems from the oracles and how much from the algorithms themselves. Concretely, most supervised QML methods assume the existence of a fast ($\log(n)$ for input dimension n complexity) data preparation oracle or procedure, a quantum equivalent of a random access memory, the qRAM [6, 7]. A closer analysis of these algorithms indeed implies that much of their power stems from this oracle, which indicates that classical algorithms can potentially achieve similar computational complexities if they are given access to such a device. The second research question of this thesis is therefore the following:

Research Question 2 (QML under the lens of RandNLA)**.** *Under the assumption of efficient sampling processes for the data for both classical and quantum algorithms, what is the comparative advantage of the latter?*

The goal of this thesis is therefore to investigate the above two challenges to supervised QML, and the resulting research questions. We thereby aim to assess the advantages and disadvantages of such algorithms from the perspective of statistical learning theory and randomised numerical linear algebra.

1.1   Synopsis of the thesis

The thesis is structured as follows. We begin in Chapter 2 with notation and mathematical preliminaries of the thesis. In Chapter 3, we discuss the first challenge and associated Research Question 1, namely the ability of quantum algorithms to generalise to data that is not present in the training set. Limitations arise through the fundamental requirement of the quantum measurement, and the error-dependency of most supervised quantum algorithms. We additionally discuss the condition number, an additional dependency in many supervised QML algorithms.

In Chapter 4 we discuss the second challenge, i.e., Research Question 2 regarding input models. For this we discuss first how to access data with a quantum computer and how these access models compare to classical approaches, in particular to randomised classical algorithms. This also includes a brief discussion about the feasibility of such memory models as well as potential advantages and disadvan-

tages. We then propose a new algorithm for Hamiltonian simulation [8], a common subroutine of many supervised QML algorithms. Next, we propose a randomised classical algorithm for the same purpose, and then use this to discuss and compare the requirements of both algorithms, and how they relate to each other. Based on these insights, we next link our results to subsequent 'dequantization' results. This allows us to finally also make claims regarding the limitations of QML algorithms that are based on such memory models (or more generally such oracles).

In Chapter 5, we argue that generative quantum machine learning algorithms that are able to generate quantum data (QQ in Fig. 1.1), are one of the most promising approaches for future QML research, and we propose a novel algorithm for the training of Quantum Boltzmann Machines (QBMs) [9] with visible and hidden nodes. We note that quantum data is here defined as data that is generated by an arbitrarily-sized quantum circuit. We derive error bounds and computational complexities for the training QBMs based on two different training algorithms, which vary based on the model assumptions.

Finally, in Chapter 6, we summarise our insights and discuss the main results of the thesis, ending with a proposal for further research.

1.2 Summary of our contributions

Our responses to the above-posed research questions are summarised below.

For Research Question 1, we obtain the following results:

- Taking into account optimal learning rates and the generalisation ability, we show that supervised quantum machine learning algorithms fail to achieve exponential speedups.

- Our analysis is based on the polynomial error-dependency and the repeated measurement that is required to obtain a classical output. We have the requirement that the (optimisation) error of the algorithm matches the scaling of the statistical (estimation) error that is inherent to all data-based algorithms and is of $O(1/\sqrt{n})$, for n being the number of samples in the data set. We observe that the performance of the algorithms in question can in practice be

worse than the fastest classical solution.

- Such concerns are important for the design of quantum algorithms but are not relevant for classical ones. The computational complexity of the latter scales typically poly-logarithmic in the error ε and therefore only introduces additional $O(\log(n))$ terms. We however acknowledge this is not generally true and it is possible to trade-off speed versus accuracy, which is for example used in early stopping. Here, one chooses to obtain a lower accuracy in order to obtain an algorithm that converges faster to the solution with the best statistical error.

- One additional challenge for many quantum algorithms is the problem of pre-conditioning, which most state-of-the-art classical algorithms such as FAL-CON [10] do take into account.

- While previous results [11] claimed that preconditioning is possible, in the quantum case, this turns out not to be true in general as, e.g. Harrow and La Placa [12] showed. Our bounds on the condition number indicate that ill-conditioned QML algorithms are even more prone to be outperformed by classical counterparts.

For Research Question 2, we obtain the following results:

- We first show that data access oracles, such as quantum random access memory [6] can be used to construct faster quantum algorithms for dense Hamiltonian simulation, and obtain an algorithm with performance (runtime) depending on the square-root of the dimension n of the input Hamiltonian, and linearly on the spectral norm, i.e., $O(\sqrt{n}\,\|H\|)$ for Hamiltonian $H \in \mathbb{C}^{n \times n}$. For a s-sparse Hamiltonian, this reduces to time $O(\sqrt{s}\,\|H\|)$, where s is the maximum number of non-zero elements in the rows of H.

- We next show how we can derive a classical algorithm for the same task, which is based on a classical Monte-Carlo sampling method called *Nyström method*. We show that for a sparse Hamiltonian $H \in \mathbb{C}^{n \times n}$, there exists an

algorithm that, with probability $1 - \delta$ approximates any chosen amplitude of
the state $e^{iHt}\psi$, for an efficiently describable state ψ in time

$$O\left(sq + \frac{t^9 \|H\|_F^4 \|H\|^7}{\varepsilon^4}\left(\log(n) + \log\frac{1}{\delta}\right)^2\right),$$

where ε determines the quality of the approximation, $\|\cdot\|_F$ is the Frobenius
norm, and q is the number of non-zero elements in ψ.

- While our algorithm is not generally efficient due to the dependency on the
  Frobenius norm, it still removes the explicit dependency on n up to logarith-
  mic factors, i.e., the system dimension. We therefore obtain an algorithm
  that only depends on the rank, sparsity, and the spectral norm of the prob-
  lem, since $\|H\|_F \leq \sqrt{r}\|H\|$, for r being the rank of H. For low-rank ma-
  trices, we therefore obtain a potentially much faster algorithm, and for the
  case $q = s = r = O(\text{polylog}(n))$ our algorithm becomes efficiently executable.
  Note that we here generally assume that N grows exponentially in the number
  of qubits in the system, i.e., we only obtain an efficient algorithm if we do not
  have an explicit n dependency.

- This result is interesting in two different ways: Firstly, it is the first known
  result that applies so-called sub-sampling methods for simulating (general)
  Hamiltonians on a classical computer. Secondly, and more important in con-
  text of this thesis, our result indicates that the advantage of so-called *quantum
  machine learning* algorithms may not be as big as promised. QML algorithms
  such as quantum principal component analysis [13], or quantum support vec-
  tor machines [14] were claimed to be efficient for sparse or low rank input
  data, and our classical algorithm for Hamiltonian simulation is efficient un-
  der similar conditions. As a corollary, we indeed show that we can efficiently
  simulate $\exp(i\rho t)$ for density matrix ρ, if we can efficiently sample from the
  rows and columns of ρ according to a certain probability distribution.

- While we did not manage to extend our results to quantum machine learn-

ing algorithms, shortly after we posted our result to the ArXiv, Ewin Tang used similar methods to 'dequantize' [15] the quantum recommendation systems algorithm by Kerenidis and Prakash [16]. While our algorithm requires the input matrix to be efficiently row-computable, Tang designed a classical memory model that allows us to sample the rows or columns of the input matrix according to their norms. Under closer inspection, these requirements are fundamentally equivalent. Furthermore, the ability to sample efficiently from this distribution is indeed similar to the ability of a quantum random access memory to prepare quantum states of the rows and columns of an input matrix. Dequantization then refers to a classical algorithm that can perform the same task as a chosen quantum algorithm with an at most polynomial slowdown. In the subsequent few months, many other algorithms were published that achieved similar results for many other quantum machine learning algorithms including quantum PCA [17], Quantum Linear Systems Algorithm [18, 19], and Quantum Semi-Definite Programming [20]. Most of these algorithms were unified recently in the framework of quantum singular value transformations [21].

A conclusion from the above results and the answers to the research questions is that the hope for exponential advantages of quantum algorithms for machine learning over their classical counterparts might be misplaced. While polynomial advantages may still be feasible, and most results currently indicate a gap, understanding the limitations of classical and quantum algorithms is still an open research question. As many of the algorithms we investigate in this thesis are of a theoretical nature, we want to mention that the ultimate performance test for any algorithm is a benchmark, and only such will give the final answers to questions of real advantage. However, this will only be possible once sufficiently large and accurate quantum computers are available, and will therefore not be possible in the near future.

As our and subsequent results indicate, quantum machine learning algorithms for classical data to date appear to allow for at most polynomial speedups if any.

We therefore turn to an area where classical machine learning algorithms might generally be inefficient: modelling of quantum distributions, and therefore quantum algorithms for quantum data. We propose a method for fully quantum generative training of quantum Boltzmann machines that in contrast to prior art have both visible and hidden units. We base our training on the quantum relative entropy objective function and find efficient algorithms for training based on gradient estimations under the assumption that Gibbs state preparation for the model Hamiltonian is efficient. One interesting feature of these results is that such generative models can in principle also be used for the state preparation, and might therefore be a useful tool to overcome qRAM-related issues.

1.3 Statement of authorship

This thesis is based on the following research articles. Notably, for many articles the order of authors is alphabetical, which is standard in theoretical computer science.

1. Danial Dervovic, Mark Herbster, Peter Mountney, Simone Severini, Naïri Usher, and Leonard Wossnig. Quantum linear systems algorithms: a primer. *arXiv preprint arXiv:1802.08227*, 2018

2. Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209):20170551, 2018

3. Chunhao Wang and Leonard Wossnig. A quantum algorithm for simulating non-sparse hamiltonians. *Quantum Information & Computation*, 20(7&8):597–615, 2020

4. Alessandro Rudi, Leonard Wossnig, Carlo Ciliberto, Andrea Rocchetto, Massimiliano Pontil, and Simone Severini. Approximating hamiltonian dynamics with the nyström method. *Quantum*, 4:234, 2020

5. Carlo Ciliberto, Andrea Rocchetto, Alessandro Rudi, and Leonard Wossnig. Fast quantum learning with statistical guarantees. *arXiv preprint arXiv:2001.10477*, 2020

6. Nathan Wiebe and Leonard Wossnig. Generative training of quantum boltzmann machines with hidden units. *arXiv preprint arXiv:1905.09902*, 2019

I have additionally co-authored the following articles that are not included in this thesis:

1. Hongxiang Chen, Leonard Wossnig, Simone Severini, Hartmut Neven, and Masoud Mohseni. Universal discriminative quantum neural networks. *arXiv preprint arXiv:1805.08654*, 2018

2. Marcello Benedetti, Edward Grant, Leonard Wossnig, and Simone Severini. Adversarial quantum circuit learning for pure state approximation. *New Journal of Physics*, 21(4):043023, 2019

3. Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019

4. Shuxiang Cao, Leonard Wossnig, Brian Vlastakis, Peter Leek, and Edward Grant. Cost-function embedding and dataset encoding for machine learning with parametrized quantum circuits. *Physical Review A*, 101(5):052309, 2020

5. I Rungger, N Fitzpatrick, H Chen, CH Alderete, H Apel, A Cowtan, A Patterson, D Munoz Ramo, Y Zhu, NH Nguyen, et al. Dynamical mean field theory algorithm and experiment on quantum computers. *arXiv preprint arXiv:1910.04735*, 2019

6. Andrew Patterson, Hongxiang Chen, Leonard Wossnig, Simone Severini, Dan Browne, and Ivan Rungger. Quantum state discrimination using noisy quantum neural networks. *arXiv preprint arXiv:1911.00352*, 2019

7. Jules Tilly, Glenn Jones, Hongxiang Chen, Leonard Wossnig, and Edward Grant. Computation of molecular excited states on ibmq using a discriminative variational quantum eigensolver. *arXiv preprint arXiv:2001.04941*, 2020

Chapter 2

Notation And Mathematical Preliminaries

2.1 Notation

We denote vectors with lower-case letters. For a vector $x \in \mathbb{C}^n$, let x_i denotes the i-th element of x. A vector is sparse if most of its entries are 0. For an integer k, let $[k]$ denotes the set $\{1, \ldots, k\}$.

For a matrix $A \in \mathbb{C}^{m \times n}$ let $A^j := A_{:,j}$, $j \in [n]$ denote the j-th column vector of A, $A_i := A_{i,:}$, $i \in [m]$ the i-th row vector of A, and $A_{ij} := A(i,j)$ the (i,j)-th element. We denote by $A_{i:j}$ the sub-matrix of A that contains the rows from i to j.

The *supremum* is denoted as sup and the *infimum* as inf. For a measure space (X, Σ, μ), and a measurable function f an essential upper bound of f is defined as $U_f^{\text{ess}} := \{l \in \mathbb{R} : \mu(f^{-1}(l, \infty)) = 0\}$, if the measurable set $f^{-1}(l, \infty)$ is a set of measure zero, i.e., if $f(x) \leq l$ for almost all $x \in X$. Then the essential supremum is defined as ess $\sup f := \inf U_f^{\text{ess}}$. We let the span of a set $S = \{v_i\}_1^k \subseteq \mathbb{C}^n$ be defined by span$\{S\} := \left\{x \in \mathbb{C}^n \,|\, \exists \{\alpha_i\}_1^k \subseteq \mathbb{C} \text{ with } x = \sum_{i=1}^k \alpha_i v_i\right\}$. The set is linearly independent if $\sum_i \alpha_i v_i = 0$ if and only if $\alpha_i = 0$ for all i. The range of $A \in \mathbb{C}^{m \times n}$ is defined by range$(A) = \{y \in \mathbb{R}^m : y = Ax \text{ for some } x \in \mathbb{C}^n\} = \text{span}(A^1, \ldots, A^n)$. Equivalently the range of A is the set of all linear combinations of the columns of A. The nullspace null(A) (or kernel ker(A)) is the set of vectors such that $Av = 0$. Given a set $S = \{v_i\}_1^k \subseteq \mathbb{C}^n$. The null space of A is null$(A) = \{x \in \mathbb{R}^b : Ax = 0\}$.

The *rank* of a matrix $A \in \mathbb{C}^{m \times n}$, rank$(A)$ is the dimension of range(A) and is equal to the number of linearly independent columns of A; Since this is equal to rank(A^H), A^H being the complex conjugate transpose of A, it also equals the number of linearly independent rows of A, and satisfies rank$(A) \leq \min\{m,n\}$. The trace of a matrix is the sum of its diagonal elements $\text{Tr}(A) = \sum_i A_{ii}$. The support of a vector supp(v) is the set of indices i such that $v_i = 0$ and we call it sparsity of the vector. For a matrix we denote the sparsity as the number of non zero entries, while row or column sparsity refers to the number of non-zero entries per row or column. A symmetric matrix A is positive semidefinite (PSD) if all its eigenvalues are non-negative. For a PSD matrix A we write $A \succeq 0$. Similarly $A \succeq B$ is the partial ordering which is equivalent to $A - B \succeq 0$.

We use the following standard norms. The Frobenius norm $\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} A_{ij}^*}$, and the spectral norm $\|A\| = \sup_{x \in \mathbb{C}^n, x \neq 0} \frac{|Ax|}{|x|}$. Note that that $\|A\|_F^2 = \text{Tr}(A^H A) = \text{Tr}(AA^H)$. Both norms are submultiplicative and unitarily invariant and they are related to each other as $\|A\| \leq \|A\|_F \leq \sqrt{n} \|A\|$.

The singular value decomposition of A is $A = U\Sigma V^H$ where U, V are unitary matrices and U^H defines the complex conjugate transpose, also called Hermitian conjugate, of U. We use throughout the thesis x^H or A^H for the Hermitian conjugate as well as for the transpose for real matrices and vectors. We denote the pseudo-inverse of a matrix A with singular value decomposition $U\Sigma V^H$ as $A^+ := V\Sigma^+ U^H$.

2.2 Matrix functional analysis

While computing the gradient of the average log-likelihood is a straightforward task when training ordinary Boltzmann machines, finding the gradient of the quantum relative entropy is much harder. The reason for this is that in general $[\partial_\theta H(\theta), H(\theta)] \neq 0$. This means that the ordinary rules that are commonly used in calculus for finding the derivative no longer hold. One important example that we will use repeatedly is Duhamel's formula:

$$\partial_\theta e^{H(\theta)} = \int_0^1 ds \, e^{H(\theta)s} \partial_\theta H(\theta) e^{H(\theta)(1-s)}. \tag{2.1}$$

This formula can be easily proven by expanding the operator exponential in a Trotter-Suzuki expansion with r time-slices, differentiating the result and then taking the limit as $r \to \infty$. However, the relative complexity of this expression compared to what would be expected from the product rule serves as an important reminder that computing the gradient is not a trivial exercise. A similar formula also exists for the logarithm as shown further below.

Similarly, because we are working with functions of matrices here we need to also work with a notion of monotonicity. We will see that for some of our approximations to hold we will also need to define a notion of concavity (in order to use Jensen's inequality). These notions are defined below.

Definition 1 (Operator monoticity). *A function f is operator monotone with respect to the semidefinite order if $0 \preceq A \preceq B$, for two symmetric positive definite operators implies, $f(A) \preceq f(B)$. A function is operator concave w.r.t. the semidefinite order if $cf(A) + (1-c)f(B) \preceq f(cA + (1-c)B)$, for all positive definite A, B and $c \in [0,1]$.*

We now derive or review some preliminary equations that we will need in order to obtain a useful bound on the gradients in the main work.

Claim 1. *Let $A(\theta)$ be a linear operator which depends on the parameters θ. Then*

$$\frac{\partial}{\partial \theta} A(\theta)^{-1} = -A^{-1} \frac{\partial \sigma}{\partial \theta} A^{-1}. \tag{2.2}$$

Proof. The proof follows straight forward by using the identity I.

$$\frac{\partial I}{\partial \theta} = 0 = \frac{\partial}{\partial \theta} A A^{-1} = \left(\frac{\partial A}{\partial \theta}\right) A^{-1} + A \left(\frac{\partial A^{-1}}{\partial \theta}\right).$$

Reordering the terms completes the proof. This can equally be proven using the Gateau derivative. □

In the following we will furthermore rely on the following well-known inequality:

Lemma 1 (Von Neumann Trace Inequality). *Let $A \in \mathbb{C}^{n \times n}$ and $B \in \mathbb{C}^{n \times n}$ with singular values $\{\sigma_i(A)\}_{i=1}^{n}$ and $\{\sigma_i(B)\}_{i=1}^{n}$ respectively such that $\sigma_i(\cdot) \leq \sigma_j(\cdot)$ if $i \leq j$.*

It then holds that

$$|\text{Tr}(AB)| \leq \sum_{i=1}^{n} \sigma(A)_i \sigma(B)_i. \tag{2.3}$$

Note that from this we immediately obtain

$$|\text{Tr}(AB)| \leq \sum_{i=1}^{n} \sigma(A)_i \sigma(B)_i \leq \sigma_{max}(B) \sum_i \sigma(A)_i = \|B\| \sum_i \sigma(A)_i. \tag{2.4}$$

This is particularly useful if A is Hermitian and PSD, since this implies $|\text{Tr}(AB)| \leq \|B\| \text{Tr}(A)$ for Hermitian A.

Since we are dealing with operators, the common chain rule of differentiation does not hold generally. Indeed the chain rule is a special case if the derivative of the operator commutes with the operator itself. Since we are encountering a term of the form $\log \sigma(\theta)$, we cannot assume that $[\sigma, \sigma'] = 0$, where $\sigma' := \sigma^{(1)}$ is the derivative w.r.t., θ. For this case we need the following identity similarly to Duhamels formula in the derivation of the gradient for the purely-visible-units Boltzmann machine.

Lemma 2 (Derivative of matrix logarithm [34])**.**

$$\frac{d}{dt} \log A(t) = \int_0^1 [sA + (1-s)I]^{-1} \frac{dA}{dt} [sA + (1-s)I]^{-1}. \tag{2.5}$$

For completeness we here include a proof of the above identity.

Proof. We use the integral definition of the logarithm [35] for a complex, invertible, $n \times n$ matrix $A = A(t)$ with no real negative

$$\log A = (A - I) \int_0^1 ds[s(A - I) + I]^{-1}. \tag{2.6}$$

From this we obtain the derivative

$$\frac{d}{dt} \log A = \frac{dA}{dt} \int_0^1 ds[s(A - I) + I]^{-1} + (A - I) \int_0^1 ds \frac{d}{dt} [s(A - I) + I]^{-1}.$$

Applying (2.2) to the second term on the right hand side yields

$$\frac{d}{dt}logA = \frac{dA}{dt}\int_0^1 ds[s(A-I)+I]^{-1}+(A-I)\int_0^1 ds[s(A-I)+I]^{-1}s\frac{dA}{dt}[s(A-I)+I]^{-1},$$

which can be rewritten as

$$\frac{d}{dt}logA = \int_0^1 ds[s(A-I)+I][s(A-I)+I]^{-1}\frac{dA}{dt}[s(A-I)+I]^{-1} \qquad (2.7)$$

$$+(A-I)\int_0^1 ds[s(A-I)+I]^{-1}s\frac{dA}{dt}[s(A-I)+I]^{-1}, \qquad (2.8)$$

by adding the identity $I = [s(A-I)+I][s(A-I)+I]^{-1}$ in the first integral and reordering commuting terms (i.e., s). Notice that we can hence just substract the first two terms in the integral which yields (2) as desired. $\qquad\square$

minimises

Chapter 3

Statistical Learning Theory

This chapter applies insights from statistical learning theory to answer the following question:

Research Question 1 (QML under the lens of SLT)**.** *Under the common assumptions of statistical learning theory, what is the performance of supervised quantum machine learning algorithms?*

The main idea in this chapter is to leverage the framework of statistical learning theory to understand how the minimum number of samples required by a learner to reach a target generalisation accuracy influences the overall performance of quantum algorithms. By taking into account well known bounds on this accuracy, we can show that quantum machine learning algorithms for supervised machine learning are unable to achieve polylogarithmic runtimes in the input dimension. Notably, the results presented here hold only for supervised quantum machine learning algorithms for which statistical guarantees are available. Our results show that without further assumptions on the problem, known quantum machine learning algorithms for supervised learning achieve only moderate polynomial speedups over efficient classical algorithms - if any. We note that the quantum machine learning algorithms that we analyse here are all based on fast quantum linear algebra subroutines [36, 37]. These in particular include quantum quantum support vector machines [14], quantum linear regression, and quantum least squares [38, 39, 40, 41, 37, 42].

Notably, the origin of the 'slow down' of quantum algorithms under the above consideration is twofold.

First, most of the known quantum machine learning algorithms have at least an inversely linear scaling in the optimisation or approximation error of the algorithm, i.e., they require a computational time $O(\varepsilon^{-\alpha})$ (for some real positive α) to return a solution $\tilde{x}$ which is ε-close in some norm to the true solution x of the problem. For example, in case of the linear systems algorithm, we obtain a quantum state $|\tilde{x}\rangle$ such that $\||\tilde{x}\rangle - |x\rangle\| \leq \varepsilon$, where $|x\rangle$ is the state that encodes the exact solution to the linear system $|x\rangle = |A^{-1}b\rangle$. This is in stark contrast to classical algorithms which typically have logarithmic dependency with respect to the error, i.e., for running the algorithm for time $O(\log(\varepsilon))$, we obtain a solution with error ε.

Second, a crucial bottleneck in many quantum algorithms is the requirement to sample in the end of most quantum algorithms. This implies generally another error, since we need to repeatedly measure the resulting quantum state in order to obtain the underlying classical result.

We note that we mainly leverage these two sources of error in the following analysis, but the extension of this to further include noise in the computation is straightforward. Indeed, noise in the computation (a critical issue in the current generation of quantum computers) could immediately be taken into account by simply adding a linear factor in terms of the error decomposition that we will encounter in Eq. 3.22. This will be a further limiting factor for near term devices as such errors need to be surpressed sufficiently in order to obtain good general bounds.

While previous research has already identified a number of caveats such as the data access [43] or restrictive structural properties of the input data, which limit the practicality of these algorithms, our insights are entirely based on statistical analysis.

As we will discuss in more detail in chapter 4, under the assumption that we can efficiently sample rows and columns of the input matrix (i.e., the input data) according to a certain distribution, classical algorithms can be shown to be nearly as efficient as these quantum machine learning algorithms [17]. We note that the

scaling of the quantum machine learning algorithms typically still achieve a high polynomial advantage compared to the classical ones.

This chapter is organised as follows. First, in Section 3.1, we will review existing results from statistical learning theory in order to allow the reader to follow the subsequent argument. Section 3.3.1 takes into account the error that is introduced by the algorithm itself. In Section 3.3.2, we then use this insight to bound the error that is induced through the sampling process in quantum mechanics. An additional dependency is typically given by the condition number of the problem, and we hence derive additional bounds for it in Section 3.3.3. Finally, in Section 3.4, we accumulate these insights and use them to analyse a range of existing supervised quantum machine learning algorithms.

3.1   Review of key results in Learning Theory

Statistical Learning Theory has the aim to statistically quantify the resources that are required to solve a learning problem [44]. Although multiple types of learning settings exist, depending on the access to data and the associated error of a prediction, here we primarily focus on supervised learning. In supervised learning, the goal is to find a function that fits a set of input-output training examples and, more importantly, guarantees also a good fit on data points that were not used during the training process. The ability to extrapolate to data points that are previously not observed is also known as *generalisation ability* of the model. This is indeed the major difference between machine learning and standard optimisation processes. Although this problem can be cast into the framework of optimisation (by optimising a certain problem instance), setting up this instance to achieve a maximum generalisation performance is indeed one of the main objectives of machine learning.

After the review of the classical part, i.e., the important points of consideration for any learning algorithm and the assumptions taken, we also analyse existing quantum algorithms, and their computational speedups within the scope of statistical learning theory.

3.1.1 Supervised Learning

We now set the stage for the analysis by defining the framework of supervised learning. Let X and Y be probability spaces with distribution ρ on $X \times Y$ from which we sample data points $x \in X$ and the corresponding labels $y \in Y$. We refer to X and Y as input set and output set respectively. Let $\ell : Y \times Y \mapsto \mathbb{R}$ be the loss function measuring the discrepancy between any two points in the input space, which is a point-wise error measure. There exist a wide range of suitable loss-functions, and choosing the appropriate one in practice is of great importance. Typical error functions are the least-squares error, $\ell_{sq}(f(x), y) := (f(x) - y)^2$ over $Y = \mathbb{R}$ for regression (generally for dense Y), or the $0 - 1$ loss $\ell_{0-1}(f(x), y) := \delta_{f(x),y}$ over $Y = \{-1, 1\}$ for classification (generally for discrete Y). The least squares loss will also use be used frequently throughout this chapter. For any hypothesis space $\mathcal{H}$ of measurable functions $f : X \mapsto Y$, $f \in \mathcal{H}$, the goal of supervised learning is then to minimise the *expected risk* or *expected error* $\mathcal{E}(f) := \mathbb{E}_{\rho}[\ell(y, f(x))]$, i.e.,

$$\inf_{f \in \mathcal{H}} \mathcal{E}(f), \quad \mathcal{E}(f) = \int_{X \times Y} \ell(f(x), y)) d\rho(x, y). \tag{3.1}$$

We hence want to minimise the expected prediction error for a hypothesis $f : X \to Y$, which is the average error with respect to the probability distribution ρ. If the loss function is measurable, then the target space is the space of all measurable functions. The space of all functions for which the expected risk is well defined is called the *target space*, and typically denoted by $\mathcal{F}$.

For many loss functions we can in practice not achieve the infimum, however, it is still possible to derive a minimizer. In order to be able to efficiently find a solution to Eq. 3.1, rather than searching over the entirety of $\mathcal{F}$, we restrict the search over a restricted hypothesis space $\mathcal{H}$, which indeed can be infinite.

In summary, a learning problem is defined by the following three components:

1. A probability space $X \times Y$ with a Borel probability measure ρ.

2. A measureable loss function $\ell : Y \times Y \mapsto [0, \infty)$.

3. A hypothesis space $\mathcal{H}$ from which we choose our hypothesis $f \in \mathcal{H}$.

The data or input spaces X can be vector spaces (linear spaces) such as $X = \mathbb{R}^d$, $d \in \mathbb{N}$, or *structured spaces*, and the output space Y can also take a variety of forms as we mentioned above.

In practice, the underlying distribution ρ is unknown and we can only access it through a finite number of observations This finite set of samples, $S_n = \{(x_i, y_i)\}_{i=1}^n$, $x_i \in X$, $y_i \in Y$, is called the training set, and we generally assume that these are sampled identically and independently according to ρ. This is given in many practical cases but it should be noted that this is not generally true. Indeed, for example for time series, subsequent samples are typically highly correlated, and the following analysis hence does not immediately hold. The assumption of independence can however be relaxed to the assumption that the data does only depend *slightly* on each other via so-called mixing conditions. Under such assumptions, most of the following results for the independent case still hold with only slight adaptations.

The results throughout this chapter rely on the following assumptions.

Assumption 1. *The probability distribution on the data space $X \times Y$ can be factorized into a marginal distribution ρ_X on X and a conditional distribution $\rho(\cdot|x)$ on Y.*

We add as a remark the observation that the probability distribution ρ can take into account a large set of uncertainty in the data. The results therefore hold for a range of noise types or partial information.

Assumption 2. *The probability distribution ρ is known only through a finite set of samples $S_n = \{x_i, y_i\}_{i=1}^n$, $x_i \in X$, $y_i \in Y$ which are sampled i.i.d. according to the Borel probability measure ρ on the data space $X \times Y$.*

3.1.2 Empirical risk minimization and learning rates

Under the above assumptions, the goal of a learning algorithm is then to choose a suitable hypothesis $f_n : X \mapsto Y$, $f_n \in \mathcal{H}$ for the minimizer of the expected risk based on the data set S_n. Empirical Risk Minimization (ERM) approaches this problem

by choosing a hypothesis that minimises the *empirical risk*,

$$\inf_{f \in \mathcal{H}} \mathcal{E}_n(f), \quad \mathcal{E}_n(f) := \frac{1}{n} \sum_{(x_i, y_i) \in S_n} \ell(f(x_i), y_i), \quad \text{(Empirical Risk)} \quad (3.2)$$

given the i.i.d. drawn data points $\{(x_i, y_i)\}_{i=1}^{n} \sim \rho^n$. Note that

$$\mathbb{E}_{\{x,y\}_{i=1}^{n} \sim \rho^n}(\mathcal{E}_n(f)) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{(x_i, y_i) \sim \rho} [\ell(f(x_i), y_i)] = \mathcal{E}(f),$$

i.e., the expectation of the empirical risk is the expected risk, which implies that we can indeed use the empirical risk as proxy for the expected risk in expectation.

While we would like to use the empirical risk as a proxy for the true risk, we also need to ensure by minimising the empirical risk, we actually find a valid solution to the underlying problem, i.e., the f_n that we find is approaching the true solution f_* which minimises the empirical risk. This requirement is termed consistency. To define this mathematically, let

$$f_n := \operatorname*{argmin}_{f \in \mathcal{H}} \mathcal{E}_n(f) \quad (3.3)$$

be the minimizer of the empirical risk, which exists under weak assumptions on $\mathcal{H}$. Then, the overall goal of a learning algorithm is to minimise the *excess risk*,

$$\mathcal{E}(f_n) - \inf_{f \in \mathcal{H}} \mathcal{E}(f), \quad \text{(Excess risk)} \quad (3.4)$$

while ensuring that f_n is *consistent* for a particular distribution ρ, i.e., that

$$\lim_{n \to \infty} \left(\mathcal{E}(f_n) - \inf_{f \in \mathcal{H}} \mathcal{E}(f) \right) = 0. \quad \text{(Consistency)} \quad (3.5)$$

Since S_n is a randomly sampled subset, we can analyse this behaviour in expectation, i.e.,

$$\lim_{n \to \infty} \mathbb{E} \left[\mathcal{E}(f_n) - \inf_{f \in \mathcal{H}} \mathcal{E}(f) \right] = 0 \quad (3.6)$$

or in probability, i.e.,

$$\lim_{n \to \infty} \text{Pr}_{\rho^n} \left[\mathscr{E}(f_n) - \inf_{f \in \mathscr{H}} \mathscr{E}(f) \right] > \varepsilon = 0 \tag{3.7}$$

for all $\varepsilon > 0$. If the above requirement holds for all distributions ρ on the data space, then we say that it is universally consistent.

However, consistency is not enough in practice, since the convergence of the risk of the empirical risk minimizer and the minimal risk could be impracticably slow.

One of the most important questions in a learning setting is therefore how fast this convergence happens, which is defined through the so-called *learning rate*, i.e., the rate of decay of the excess risk. We assume that this scales somewhat with respect to n, as for example

$$\mathbb{E} \left[\mathscr{E}(f_n) - \inf_{f \in \mathscr{F}} \mathscr{E}(f) \right] = O(n^{-\alpha}). \tag{3.8}$$

This speed of course has a practical relevance and hence allows us to compare different algorithms. The sample complexity n must depend on the error we want to achieve, and in practice we can therefore define it as follows: For a distribution ρ, $\forall \delta, \varepsilon > 0$, there exists a $n(\delta, \varepsilon)$ such that

$$\text{Pr}_{\rho^n} \left[\mathscr{E}(f_{n(\varepsilon, \delta)}) - \inf_{f \in \mathscr{F}} \mathscr{E}(f) \leq \varepsilon \right] \geq 1 - \delta. \tag{3.9}$$

This $n(\delta, \varepsilon)$ is called the sample complexity.

One challenge of studying our algorithm's performance through the excess risk is that we can generally not assess it. We therefore need to find a way estimate or bound the excess risk solely based on the hypothesis f_n and the empirical risk $\mathscr{E}_n$.

Let us for this assume the existence of a minimizer

$$f_* := \inf_{f \in \mathscr{H}} \mathscr{E}(f) \tag{3.10}$$

over a suitable Hypothesis space $\mathscr{H}$.

We can then decompose the excess risk as

$$\mathscr{E}(f_n) - \mathscr{E}(f_*) = \mathscr{E}(f_n) - \mathscr{E}_n(f_n) + \mathscr{E}_n(f_n) - \mathscr{E}_n(f_*) + \mathscr{E}_n(f_*) - \mathscr{E}(f_*). \quad (3.11)$$

Now observe that $\mathscr{E}_n(f_n) - \mathscr{E}_n(f_*) \leq 0$, we immediately see that we can bound this quantity as

$$\mathscr{E}(f_n) - \mathscr{E}(f_*) = \mathscr{E}(f_n) - \inf_{f \in \mathscr{H}} \mathscr{E}(f) \leq 2 \sup_{f \in \mathscr{H}} |\mathscr{E}(f) - \mathscr{E}_n(f)|, \quad (3.12)$$

which implies that we can bound the error in terms of the so-called *generalisation error* $\mathscr{E}(f) - \mathscr{E}_n(f)$.

We therefore see that we can study the convergence of the excess risk in terms of the generalisation error. Controlling the generalisation error is one of the main objectives of statistical learning theory.

A fundamental result in statistical learning theory, which is often referred as the fundamental theorem of statistical learning, is the following:

Theorem 1 (Fundamental Theorem of Statistical Learning Theory [45, 46, 44])**.** *Let $\mathscr{H}$ be a suitably chosen Hypothesis space of functions $f : X \mapsto Y$, $X \times Y$ be a probability space with a Borel probability measure ρ and a measurable loss function $ell : Y \times Y \mapsto [0, \infty)$, and let the empirical risk and risk be defined as in Eq. 3.1 and Eq. 3.2 respectively. Then, for every $n \in \mathbb{N}$, $\delta \in (0, 1)$, and distributions ρ, with probability $1 - \delta$ it holds that*

$$\sup_{f \in \mathscr{H}} |\mathscr{E}_n(f) - \mathscr{E}(f)| \leq \Theta\left(\sqrt{\frac{c(\mathscr{H}) + \log(1/\delta)}{n}}\right), \quad (3.13)$$

where $c(\mathscr{H})$ is a measure of the complexity of $\mathscr{H}$ (such as the VC dimension, covering numbers, or the Rademacher complexity [47, 44]).

We note that lower bounds for the convergence of the empirical to the (expected) risk are much harder to obtain in general, and generally require to fix the underlying distribution of the data. We indeed need for the following proof a lower bound on this complexity which is not generally available. However, for the specific

problems we are treating here this holds true and lower bounds do indeed exist. For simplicity, we will rely on the above given bound, as the other factors that occur in these bounds only play a minor role here.

3.1.3 Regularisation and modern approaches

The complexity of the hypothesis space, $c(\mathscr{H})$ in Eq. 3.13 relates to the phenomenon of *overfitting*, where a large hypothesis space results in a low training error on the empirical risk, but performs poorly on the true risk.

In the literature, this problem is addressed with so-called *regularisation techniques*, which are able to limit the size of the Hypothesis space and thereby its complexity, in order to avoid overfitting the training dataset.

A number of different regularisation strategies have been proposed in the literature [45, 48, ?], including the well-established Tikhonov regularisation, which directly imposes constraints on the hypotheses class of candidate predictors.

From a computational perspective, Tikhonov regularisation, and other similar approaches compute a solution for the learning problem by optimising a constraint objective (i.e., the empirical risk with an additional regularisation term). The solution is obtained by a sequence of standard linear algebra operations such as matrix multiplication and inversion. Since the standard matrix inversion time is $\mathscr{O}(n^3)$ for a $n \times n$ square matrix, most of the solutions such as for GP or SVM, can be found in $\mathscr{O}(n^3)$ computational time for n data points. Notably, improvements to this runtime exist based on exploiting sparsity, or trading an approximation error against a lower computational time. Additionally, the time to solution typically depends on the conditioning of the matrix, which therefore can be lowered by using preconditioning methods.

Regularisation is today widely used, and has led to many popular machine learning algorithms such as Regularised Least Squares [47], Gaussian Process (GP) Regression and Classification [49], Logistic Regression [48], and Support Vector Machines (SVM) [45]. All the above mentioned algorithms fall under the same umbrella of kernel methods [50].

To further reduce the computational cost, modern methods leverage on the fact

that regularisation can indeed be applied implicitly through incomplete optimisation or other forms of approximation. These ideas have been widely applied in the so-called *early stopping* approaches which are today standardly used in practice. In early stopping, one only performs a limited number of steps of an iterative optimisation algorithm, typically in gradient based optimisation. It can indeed be shown for convex functions, that this process avoids overfitting the training set (i.e., maintains an optimal learning rate), while the computational time is drastically reduced. All regularisation approaches hence achieve a lower number of required operations, while maintaining similar or the same generalisation performance of approaches such as Tikhonov regularisation [?], in some cases provably.

Other approaches include the *divide and conquer* [51] approach or Monte-Carlo sampling (also-called sub-sampling) approaches. While the former is based on the idea of distributing partitions of the initial training data, training different predictors on the smaller problem instances, and then combining individual predictors into a joint one, the latter achieves a form of dimensionality reduction through sampling a subset of the data in a specific manner. The most well-known sub-sampling methods are random features [52] and so called Nyström approaches [53, 54].

In both cases, computation benefits from parallelisation and the reduced dimension of the datasets while similarly maintaining statistical guarantees (e.g.,[55]).

For all the above mentioned training methods, the computational times can typically be reduced from the $\mathscr{O}(n^3)$ of standard approaches to $\widetilde{\mathscr{O}}(n^2)$ or $\widetilde{\mathscr{O}}(nnz)$, where *nnz* is the number of non-zero entries in the input data (matrix), while maintaining optimal statistical performance.

Since standard regularisation approaches can trivially be integrated into quantum algorithms - such as regularised least squares - certain methods appear not to work in the quantum algorithms toolbox.

For example, preconditioning, as a tool to reduce the condition number and make the computation more efficient appears not to have an efficient solution yet in the quantum setting [12]. Therefore, more research is required to give a full picture

of the power and limitations of algorithms with respect to all parameters. Here we offer a brief discussion of the possible effects of inverting badly conditioned matrices and how typical cases could affect the computational complexity, i.e., the asymptotic runtime of the algorithm.

3.2 Review of supervised quantum machine learning algorithms

The majority of proposed supervised quantum machine learning algorithms are based on fast linear algebra operations. Indeed, most quantum machine learning algorithms that claim an exponential improvement over classical counterparts are based on a fast quantum algorithm for solving linear systems of equations [38, 39, 14, 40, 41, 37, 42]. This widely used subroutine is the HHL algorithm, a quantum linear system solver [36] (QLSA), which is named after its inventors Harrow, Hassidim, and Lloyd. The HHL algorithm takes as input the normalised state $|b\rangle \in \mathbb{R}^n$ and a $s(A)$-sparse matrix $A \in \mathbb{R}^{n \times n}$, with spectral norm $\|A\| \leq 1$ and condition number $\kappa = \kappa(A)$, and returns as an output a quantum state $|\tilde{w}\rangle$ which encodes an approximation of the normalised solution $|w\rangle = |A^{-1}b\rangle \in \mathbb{R}^n$ for the linear system $Aw = b$ such that

$$\| |\tilde{w}\rangle - |w\rangle \| \leq \gamma, \tag{3.14}$$

for error parameter γ. Note that above we assumed that the matrix is invertible, however, the algorithm can in practice perform the Moore-Penrose inverse (also known as pseudoinverse), which is defined for arbitrary $A \in \mathbb{R}^{n \times m}$ by $A^+ := (A^H A)^{-1} A^H$, and using the singular value decomposition of $A = U \Sigma V^H$, we hence have

$$A^+ = (V \Sigma^2 V^H)^{-1} V \Sigma U^H = V \Sigma^{-1} U^H, \tag{3.15}$$

such that $A^+ A = V V^H = I_m$.

The currently best known quantum linear systems algorithm, in terms of com-

putational complexity, runs in

$$O(\|A\|_F \, \kappa \, \text{polylog}(\kappa, n, 1/\gamma)), \tag{3.16}$$

time [37], where $\|A\|_F \leq \sqrt{n} \|A\|_2 \leq \sqrt{n}$ is the Frobenius norm of A and κ its condition number. As we will discuss in more detail in Chapter 4 on randomised numerical linear algebra, such computations can also be done exponentially faster compared to known classical algorithms using classical randomised methods in combination with a quantum-inspired memory structure, by taking advantage of aforementioned Monte-Carlo sampling methods [56, 17, 18, 19, 21] if the data matrix (A) is of low rank. We note, that for full-rank matrices, an advantage is however still possible.

Before analysing the supervised machine learning algorithms with the above discussed knowledge from statistical learning theory, we will first recapitulate the quantum least squares algorithm (linear regression) and the quantum support vector machine (SVM). Throughout this chapter, we use the least squares problem as a prototypical case to study the behaviour of QML algorithms, but the results extend trivially to the quantum SVM and many other algorithms. In Section 3.4 we also summarise the computational complexities taking into account the statistical guarantees for all other algorithms and hence give explicit bounds.

3.2.1   Recap: Quantum Linear Regression and Least Squares

The least squares algorithm minimises the empirical risk with respect to the quadratic loss

$$\ell^{LS}(f(x), y) = (f(x) - y)^2, \tag{3.17}$$

for the hypothesis class of linear functions

$$\mathscr{H} := \{f : X \mapsto Y \,|\, \exists w \in \mathbb{R}^d : f(x) = w^H x\}, \tag{3.18}$$

with input space $\mathbb{R}^d$ and outputspace $\mathbb{R}$. Given input-output samples $\{x_i, y_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$, and $y_i \in \mathbb{R}$. The empirical risk is therefore given by

$$\mathscr{E}_n(f) := \frac{1}{n} \sum_{i=1}^n \left(w^H x_i - y_i\right)^2. \tag{3.19}$$

the least squares problem seeks to find a vector w such that

$$w = \operatorname{argmin}_{w \in \mathbb{R}^d} \left(\|y - Xw\|_2^2\right), \tag{3.20}$$

where $y \in \mathbb{R}^n$ and $X \in \mathbb{R}^{n \times d}$. The closed form solution of this problem is then given by $w = (X^H X)^{-1} X^H y$, and we can hence reformulate this again into a linear systems problem of the form $Aw = b$, where $A = (X^H X)$ and $b = X^H y$. We obtain the solution then by solving the linear system $w = A^{-1}b$.

Since several quantum algorithms for linear regression and in particular least squares have been proposed [38, 39, 40, 41, 37, 42], which all result in a similar scaling (taking into account the most recent subroutines), we will in the following analysis use the best known result for the quantum machine learning algorithm. All approaches have in common that they make use of the quantum linear system algorithm to convert the state $|\xi\rangle = |X^H y\rangle$ into the solution $|\tilde{w}\rangle = |(X^H X)^{-1} \xi\rangle$. The fastest known algorithm [37], indeed solves (regularised) least squares or linear regression problem in time

$$O\left(\|A\|_F \, \kappa \, \text{polylog}(n, \kappa, 1/\gamma)\right),$$

where κ^2 is the condition number of $X^H X$ or XX^H respectively, and $\gamma > 0$ is an error parameter for the approximation accuracy. Notably, this algorithm precludes a physical measurement of the resulting vector $|\tilde{w}\rangle$, since this would immediately imply a complexity of

$$O\left(\|A\|_F \, \kappa/\gamma \, \text{polylog}(n, \kappa, 1/\gamma)\right).$$

In that sense, the algorithm does solve the classical least squares problem only to

a certain extent as the solution is not accessible in that time. Indeed it prepares a quantum state $|\tilde{w}\rangle$ which is γ-close to $|w\rangle$, i.e.,

$$\left\| |\tilde{w}\rangle - |w\rangle \right\| \leq \gamma,$$

and in order to recover it, we would need to take up to $O(n \log(n))$ samples. In the current form, we can however immediately observe that the Frobenius norm dependency implies that the algorithm is efficient if X is low-rank (but no necessarily non-sparse). As for all of the supervised quantum machine learning algorithms for classical input data, the quantum least squares solver requires a quantum-accessible data structure, such as a qRAM.

Notably, it is assumed that $\frac{1}{\kappa^2} \leq \left\| X^H X \right\| \leq 1$. The output of the algorithm is then a quantum state $|\tilde{w}\rangle$, such that $\left\| |\tilde{w}\rangle - |w\rangle \right\| \leq \varepsilon$, where $|w\rangle$ is the true solution.

We note that other linear regression algorithms based on sample-based Hamiltonian simulation are possible [13, 57], which result in different requirements. Indeed, for these algorithms we need to repeatedly prepare a density matrix, which is a normalised version of the input data matrix. While this algorithm has generally worse dependencies on the error γ, it is independent of the Frobenius norm [39]. The computational complexity in this case is

$$O(\kappa^2 \gamma^{-3} \text{polylog}(n)),$$

which can likely be improved to $O(\kappa \gamma^{-3} \text{polylog}(n, \kappa, \gamma^{-1}))$.

However, since our analysis will indeed show that a higher polynomial dependency will incur a worse runtime once we take statistical guarantees into account, we will use the algorithm in the following which has the lowest dependency. Notably, the error dependency is in either case polynomial if we require a classical solution to be output.

As a further remark, other linear regression or least squares quantum algorithms exist [40, 42], but we will not include these here as our results can easily be extended to these as well.

Next, we will also recapitulate the quantum support vector machine.

3.2.2 Recap: Quantum Support Vector Machine

The second prototypical quantum machine learning algorithm which we want to recapitulate is the quantum least-squares support vector machine [14] (qSVM). As we will see, the procedure for the qSVM is similar to the quantum least squares approach, and therefore results in very similar runtimes. The qSVM algorithm is calculating the optimal separating hyperplane by solving again a linear system of equations. For n points $S_n = \{(x_i, y_i)\}_{i=1}^n$ with $x_i \in \mathbb{R}^d$, $y_i = \{\pm 1\}$, and again assuming that we can efficiently prepare states corresponding to the data vectors, then the least-squares formulation of the solution is given by the linear system of the form

$$\begin{pmatrix} 0 & \vec{1}^H \\ \vec{1} & K + \delta^{-1}I \end{pmatrix} \begin{pmatrix} w_0 \\ w \end{pmatrix} = \begin{pmatrix} 0 \\ y \end{pmatrix}, \tag{3.21}$$

where $K_{ij} = x_i^H x_j$ (or $K_{ij} = \phi(x_i)^H \phi(x_j)$ respectively for a non-linear features) is the kernel matrix, $y = (y_1, \ldots, y_n)^H$, $\vec{1}$ is the all-ones vector, and δ is a user specified parameter. We note that certain authors argue that a least square support vector machine is not truly a support vector machine, and their practical use highly restricted. The additional row and column in the matrix on the left hand side arise because of a non-zero offset. Notably, $w^H x + w_0 > 1$ or $w^H x + w_0 < -1$, with $w^H x = \sum_{j=1}^n w_j x_j$ determines the hyperplanes. The solution is hence obtained by solving the linear systems using the HHL algorithm based on the density matrix exponentiation [13] method previously mentioned. The only adaptation which is necessary is to use the normalised Kernel $\hat{K} = K/\text{tr}(K)$. However, since the smallest eigenvalues of $\hat{K}$ will be of $O(1/n)$ due to the normalisation, the quantum SVM algorithm truncates the eigenvalues which are below a certain threshold δ_K, s.t., $\delta_K \leq |\lambda_i| \leq 1$, which results in an effective condition number $\kappa_{eff} = 1/\delta_K$, thereby effectively implementing a form of spectral filtering.

The runtime of the quantum support vector machine is given by

$$O(\kappa_{eff}^3 \gamma^{-3} \text{polylog}(nd, \kappa, 1/\gamma)),$$

and outputs a state $|\tilde{w}_n\rangle$ that approximates the solution $w_n := [w_0, w]^H$, such that $\||\tilde{w}_n\rangle - |w_n\rangle\| \leq \gamma$. Similar as for the least squares algorithm, we cannot retrieve the parameters without an overhead, and the quantum SVM therefore needs to perform immediate classification.

3.3 Analysis of quantum machine learning algorithms

The quantum algorithms we analyse throughout this chapter rely on a range of parameters, which include the input dimension n, which corresponds to the number of data points in a sample (the dimension of the individual data point is typically small so we focus on this part), the error of the algorithm with respect to the final prediction γ, and the condition number κ of the input data matrix. Our main objective is to understand the performance of these algorithms if we want to achieve an overall generalisation error of $\Theta(n^{-1/2})$.

To start, we therefore first need to return to our previous assessment of the risk, and use in the following a standard error decomposition. Let f be a hypothesis, and let $\mathscr{F}$ is the space of all measurable functions $f : X \mapsto Y$. We define by $\mathscr{E}^* := \inf_{f \in \mathscr{F}} \mathscr{E}(f)$ the Bayes risk, and want to limit the distance $\mathscr{E}(f) - \mathscr{E}^*$. Let now $\mathscr{E}_{\mathscr{H}} := \inf_{f \in \mathscr{H}} \mathscr{E}(f)$, i.e., the best risk attainable by any function in the hypothesis space $\mathscr{H}$, where we assume in the following for simplicity that $\mathscr{E}_{\mathscr{H}}$ always admits a minimizer $f_{\mathscr{H}} \in \mathscr{H}$. Note, that it is possible to remove this assumption by leveraging regularisation. We then decompose the error as:

$$\mathscr{E}(f) - \mathscr{E}^* = \underbrace{\mathscr{E}(f) - \mathscr{E}(\hat{f})}_{\text{Optimisation error}} + \underbrace{\mathscr{E}(\hat{f}) - \mathscr{E}_{\mathscr{H}}}_{\text{Estimation error}} + \underbrace{\mathscr{E}_{\mathscr{H}} - \mathscr{E}^*}_{\text{Irreducible error}} \tag{3.22}$$

$$= \xi + \Theta(1/\sqrt{n}) + \mu. \tag{3.23}$$

The first term in Eq. 3.22 is the so-called *optimisation error* which indicates how good the optimisation procedure which generates f is, with respect to the actual minimum (infimum) of the empirical risk. This error stems from the approximations

an algorithm typically makes, and relates to the γ in previous sections. The optimi-sation error can result from a variety of approximations, such as a finite number of steps in an iterative optimisation process or a sample error introduced through a non-deterministic process. This error is discussed in detail in Section 3.3.1. The second term is the *estimation error* which is due to taking the empirical risk as a proxy for the true risk by using samples from the distribution ρ. This can be bound by the generalisation bound we discussed in Eq. 3.13. The last term is the *irreducible error* which measures how well the hypothesis space describes the problem. If the true solution is not in our hypothesis space, there will always be an irreducible error that we indicate with the letter μ. If $\mu = 0$, i.e., irreducible error is zero, then we call $\mathcal{H}$ is universal. For simplicity, we assume here that $\mu = 0$, as it also will not impact the results of this paper much.

From the error decomposition in Eq. 3.22 we see that in order to achieve the best possible generalisation error overall, we need to make sure that the different error contributions are of the same order. We therefore in particular need to ensure that the optimisation error matches the scaling of the estimation error. Since for most known classical algorithms, with the exceptions of e.g., Monte-Carlo algo-rithms, the optimisation error typically scales with $O(\log(1/\varepsilon))$ and matching the bounds is usually trivial. However, many quantum algorithms,including some of the quantum linear regression and least squares algorithms we discussed in the previous section (e.g. [14, 39]), have a polynomial dependency on the optimisation error. In the next section we discuss the implications of matching the bounds, and how they affect the algorithms computational complexity.

Notably, other quantum algorithms have only a polylogarithmic error depen-dency, such as [37], and therefore the error matching does not impose any critical slowdown. In these cases, however, we will see that quantum algorithms argument still cannot achieve a polylogarithmic runtime in the dimension of the training set due to the error resulting from the finite sampling process that is required to extract a classical output from a given quantum state.

Finally, to take into account all dependencies of the quantum algorithms, we

also analyse the condition number. Here, we show that with high probability the condition number has a polynomial dependency on the number of samples in the training set as well, which therefore indicates a certain scaling of the computational complexity.

We do the analysis in the following exemplary for the least squares case, and summarise the resulting computational complexities of a range of supervised quantum machine learning algorithms next to the classical ones then in Fig. 3.1.

3.3.1 Bound on the optimisation error

As previously mentioned, we will use the quantum least squares algorithms [38, 39, 40] as an example case to demonstrate how the matching of the error affects the algorithm. The results we obtain can easily be generalised to other algorithms and instances, and in particularly hold for all kernel methods. As we try to remain general, we will do the analysis with a general algorithm with the computational complexity

$$\Omega\left(n^{\alpha}\gamma^{-\beta}\kappa^{c}\log(n)\right) \tag{3.24}$$

We show that in order to have a total error that scales as $n^{-1/2}$, the quantum algorithm will pick up a polynomial n-dependency.

The known quantum least squares algorithms have a γ error guarantee for the final output state $|\tilde{w}\rangle$, i.e., $\||w\rangle - |\tilde{w}\rangle\|_2 \leq \gamma$, where $|w\rangle$ is the true solution. The computational complexity (ignoring all but the error-dependency), is for all algorithms of the form $O(\gamma^{-\beta})$ for some β, for example [39] with $\beta = 3$, or [58] with $\beta = 4$.

Since the quantum algorithms require the input data matrix to be either Hermitian or encoded in a larger Hermitian matrix, the dimensionality of the overall matrix is $n + d$ for n data points in $\mathbb{R}^d$. For simplicity, we here assume that the input matrix is a $n \times n$ Hermitian matrix, and neglect this step. In order to achieve the best possible generalisation error, as discussed previously, we want to match the errors of the incomplete optimisation to the statistical ones. In the least squares setting, $\tilde{w} = w_{\gamma,n}$ is the output of the algorithm corresponding to the optimal parameters fit-

ted to the S_n data points, which exhibits at most a γ-error. Therefore, $\tilde{w}$ corresponds to the estimator $f_{\gamma,n}$ in the previous notation that we saw in Eq. 3.22 (and Eq. 3.12). Concretely, we can see that the total error of an estimator $f_{\gamma,n}$ on n data points with precision ε is given by

$$
\begin{aligned}
\mathscr{E}(f_{\gamma,n}) - \mathscr{E}(f_n) = \\
= \underbrace{\mathscr{E}(f_{\gamma,n}) - \mathscr{E}_n(f_{\gamma,n})}_{\text{generalisation error}} + \underbrace{\mathscr{E}_n(f_{\gamma,n}) - \mathscr{E}_n(f_\gamma)}_{\text{Optimisation error}} + \underbrace{\mathscr{E}_n(f_\gamma) - \mathscr{E}(f_\gamma)}_{\text{generalisation error}} \\
= \Theta(n^{-1/2}) + \underbrace{\mathscr{E}_n(f_{\gamma,n}) - \mathscr{E}_n(f_\gamma)}_{\text{Optimisation error}},
\end{aligned}
\tag{3.25}
$$

where the first contribution is a result of Eq. 3.13, i.e., the generalisation performance, and the second comes from the error of the quantum algorithm, which we will show next. In order to achieve the best statistical performance, which means to achieve the lowest generalisation error, the algorithmic error must scale at worst as the worst statistical error. We will next show that the optimisation error of a quantum algorithm in terms of the prediction results in a γ error, which is inherited from weights $|\tilde{w}\rangle$, which the quantum algorithm produces. Recalling, that in least squares classification is performed via the inner product, i.e.,

$$
y_{pred} := \tilde{w}^H x,
\tag{3.26}
$$

for model $\tilde{w}$ and data point x which corresponds to $f_{\gamma,n}^H(x)$ in the general notation. This then will result in the expected risk of the estimator $f_{\gamma,n}$ to be

$$
\mathscr{E}_n(f_{\gamma,n}) = \frac{1}{n} \sum_{i=1}^{n} \left(\tilde{w}^H x_i - y_i \right)^2.
\tag{3.27}
$$

Therefore, assuming the output of the quantum algorithm is a state $\tilde{w}$, while the exact minimizer of the empirical risk is w, s.t., $\| |\tilde{w}\rangle - |w\rangle \|_2 \leq \gamma$, and assuming that

$|X|$ and $|Y|$ are bounded, then we find that

$$
\begin{aligned}
|\mathscr{E}_n(f_{n,\gamma}) - \mathscr{E}_n(f_n)| &\leq \frac{1}{n} \sum_{i=1}^{n} \left| \left(\tilde{w}^H x_i - y_i\right)^2 - \left(w^H x_i - y_i\right)^2 \right| \\
&\leq \frac{1}{n} \sum_{i=1}^{n} L \left| (\tilde{w} - w)^H x_i \right| \\
&\leq \frac{1}{n} \sum_{i=1}^{n} L \|\tilde{w} - w\|_2 \|x_i\|_2 \leq k \cdot \gamma = O(\gamma),
\end{aligned}
\tag{3.28}
$$

where $k > 0$ is a constant, and we used Cauchy-Schwartz, and the fact that that for the least-square it holds that

$$
|\ell^{LS}(f(x_i), y_i) - \ell^{LS}(f(x_j), y_j)| \leq L|(f(x_i) - y_i) - (f(x_j) - y_j)|,
\tag{3.29}
$$

since $|X|$, and $|Y|$ bounded.

A few remarks. In the learning setting the number of samples is fixed, and hence cannot be altered, i.e., the statistical error (generalisation error) is fixed to $\Theta(1/\sqrt{n})$, and the larger n is taken, the better the guarantees we are able to obtain for future tasks. Therefore, it is important to understand how we can reduce the other error contributions in Eq. 3.25 in order to guarantee that we have the lowest possible overall error, or accuracy.

To do so, we match the error bounds of the two contributions, so that the overall performance of the algorithm is maximised, which means that the optimisation error should not surpass the statistical error. We hence set $\gamma = n^{-1/2}$, and see that the overall scaling of the algorithm will need be of the order $O\left(n^{\beta/2}\right)$, ignoring again all other contributions. To take a concrete case, for the algorithm in [39] the overall runtime is then of at least $O(n^{3/2})$. The overall complexity of the algorithm then has the form

$$
\Omega\left(n^{\alpha} n^{\beta/2} \log(n) \kappa^c\right)
\tag{3.30}
$$

for some constant c, β, α

This straightforward argument from above can easily be generalised to arbitrary kernels by replacing the input data x with feature vectors $\phi(x)$, where $\phi(\cdot)$ is a

chosen feature map.

We have so far only spoken about algorithms which naturally have a polynomial $1/\gamma$-dependency. However, as we previously mentioned, not all quantum algorithms have such an error. For algorithms which only depend polylogarithmically on $1/\gamma$, however, the quantum mechanical nature will incur another polynomial n dependency as we will see next.

3.3.2 Bounds on the sampling error

So far we have ignored any error introduced by the measurement process. However, we will always need to compute a classical estimate of the output of the quantum algorithm, which is based on a repeated sampling of the output state. As this is an inherent process which we will need to perform for any quantum algorithm, the following analysis applies to any QML algorithm with classical output. Since we estimate the result by repeatedly measuring the final state of our quantum computation in a chosen basis, our resulting estimate for the desired output is a random variable. It is well known from the central limit theorem, that the sampling error for such a random variable scales as $O(1/\sqrt{m})$, where m is the number of independent measurements. This is known as the standard quantum limit or the so-called shot-noise limit. Using so-called quantum metrology it is sometimes possible to overcome this limit and obtain an error that scales with $1/m$. This however poses the ultimate limit to measurement precision which is a direct consequence of the Heisenberg uncertainty principle [6, 7].

Therefore, any output of the quantum algorithm will have a measurement error τ. Let us turn back to our least squares quantum algorithm. It produces a state $|\tilde{w}\rangle$ which is an approximation to the true solution $|w\rangle$. Using techniques such as quantum state tomography we can produce a classical estimate $\hat{w}$ of the vector $\tilde{w}$ with accuracy

$$\|\tilde{w} - \hat{w}\|_2 \leq \tau = \Omega(1/m), \tag{3.31}$$

where m is the number of measurements performed. If $y = w^H x$ is the error-free (ideal) prediction, then we can hence only produce an approximation $\hat{y} = \hat{w}^H x$, such

that

$$|y - \hat{y}| = |w^H x - \hat{w}^H x| \tag{3.32}$$

$$\leq \|w - \tilde{w} + \tau\| \, \|x\| \tag{3.33}$$

$$\leq (\gamma + \tau) \, \|x\|, \tag{3.34}$$

using again Cauchy-Schwartz. Similar to the previous approach, we need to make sure that the contribution coming from the measurement error scales at most as the worst possible generalisation error, and hence set $\tau = n^{-1/2}$. From this, we immediately see that any quantum machine learning algorithm which is to reach optimal generalisation performance, will require a number of $m = \Omega(n^{1/2})$ repetitions, which is hence a lower bound for all supervised quantum machine learning algorithms. For algorithms which do not take advantage of forms of advanced quantum metrology, this might even be $\Omega(n)$.

Putting things together, we therefore have a scaling of any QML algorithm of

$$\Omega\left(n^{\alpha + (1+\beta)/2} \log(n) \kappa^c\right), \tag{3.35}$$

which for the state-of-the-art quantum algorithm for quantum least squares [37] result in

$$\Omega\left(n^2 \kappa \log(n)\right). \tag{3.36}$$

In order to determine the overall complexity, we hence only have one parameter left: κ. However, already now we observe that the computational complexity is similar or even worse compared to the best classical machine learning algorithms.

3.3.3 Bounds on the condition number

In the following we will do the analysis of the last remaining depedency of the quantum algorithms. The condition number. Let the condition number dependency of the QML algorithm again be given by κ^c for some constant $c \in \mathbb{R}^+$. Note that the best known result has a $c = 1$ dependency, ignoring logarithmic dependencies. We can think of the following three scenarios for the condition number.

1. Best case scenario: In the best case setting, the condition number is one or sufficiently close to one. This is the lower bound and can only ever happen if the data is full rank and all the eigenvalues are of very similar size, i.e., $\lambda_i \approx \lambda_j$ for all i, j. However, for such cases, it would be questionable whether a machine learning algorithm would be useful, since this would imply that the data lacks any strong signal. In these cases the quantum machine learning algorithms could be very fast and might give a quantum advantage if the *n*-scaling due to the error-dependency is not too high.

2. Worst case scenario: On the other extreme, the condition number could be infinite, as could be the case for very badly conditioned matrices with smallest eigenvalues approaching 0. This can be the case if we have one or a few strong signals (i.e., eigenvalues which are closer to 1), and a small additional noise which results in the smallest eigenvalues being close to 0. Such ill conditioned systems do indeed occur in practice, but can generally be dealt with by using spectral-filtering or preconditioning methods, as for example discussed in [59]. Indeed, the quantum SVM [14] or the HHL algorithm [36] do or can readily make use of such methods. Concretely they do only invert eigenvalues which are above a certain threshold. This hence gives a new, effective condition number $\kappa_{eff} = \sigma_{max}/\sigma_{threshold} \leq 1/\sigma_{threshold}$ which is typically way smaller compared to the actual κ, and makes algorithms practically useful. However, it should be noted that quantum algorithms which perform such steps need to be compared against corresponding classical methods. Note, that such truncations (filters) typically introduce an error, which then needs to be taken into account separately. Having covered these two extreme scenarios, we can now focus on a typical case.

3. A plausible case: While the second case will appear in practice, these bounds give little insight into the actual performance of the quantum machine learning algorithms, since we cannot infer any scaling of κ from them. However, for kernel based methods, we can derive a plausible case which can give us some intuition of how bad the κ-scaling typically can be. We will in the

following show that with high probability, the condition number for a kernel method can have a certain *n*-dependency. Even though this result gives only a bound in probability, it is a plausible case with concrete *n*-dependency (the dimension of the input matrix) rather than the absolute worst case of $\kappa = \infty$, which gives an impractical upper bound. As a consequence, a quantum kernel method which scales as $O(\kappa^3)$ could pick up a factor of $O(n\sqrt{n})$ in the worst case which has the same complexity as the classical state of the art.

In the following, we now prove a lower bound for the condition number of a covariance or kernel matrix assuming that we have at least one strong signal in the data. The high level idea of this proof is that the sample covariance should be close to the true covariance with increasing number of samples, which we can show using concentration of measure. Next we use that the true covariance is known to have converging eigenvalues, as it constitutes a converging series. This means we know that the *k*-th eigenvalue of the true covariance must have an upper bound in terms of its size which is related to *k*. Since we also know that the eigenvalues of the two matrices will be close, and assuming that we have a few strong signals (i.e., $O(1)$ large eigenvalue), we can then bound the condition number as the ratio of the largest over the smallest eigenvalue.

For the following analysis, we will first need to recapitulate some well known results about Mercer's kernels, which can be found e.g., in [47]. If $f \in L^2_\nu(X)$ is a function in the Hilbert space of square integrable functions on X with Borel measure ν, and $\{\phi_1, \phi_2, \ldots\}$ is a Hilbert basis of $L^2_\nu(X)$, f can be uniquely written as $f = \sum_{k=1}^{\infty} a_k \phi_k$, and the partial sums $\sum_{k=1}^{N} a_k \phi_k$ converge to f in $L^2_\nu(X)$. If this convergence holds in $C(X)$, the space of continuous functions on X, we say that the series converges uniformly to f. If furthermore $\sum_k |a_k|$ converges, then we say that the series $\sum_k a_k$ converges absolutely. Let now $K : X \times X \to \mathbb{R}$ be a continuous function. Then the linear map

$$L_K : L^2_\nu(X) \to C(X)$$

given by the following integral transform

$$(L_K f)(x) = \int K(x, x') f(x') d\nu(x')$$

is well defined. It is well known that the integral operator and the kernel have the following relationship:

Theorem 2 ([47][Theorem 1, p. 34; first proven in [60]). *] Let $\mathcal{X}$ be a compact domain or a manifold, ν be a Borel measure on $\mathcal{X}$, and $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a Mercer kernel. Let λ_k be the kth eigenvalue of L_K and $\{\phi_k\}_{k \geq 1}$ be the corresponding eigenvectors. Then, we have for all $x, x' \in \mathcal{X}$*

$$K(x, x') = \sum_{k=1}^{\infty} \lambda_k \phi_k(x) \phi_k(x'),$$

where the convergence is absolute (for each $x, x' \in \mathcal{X} \times \mathcal{X}$) and uniform (on $\mathcal{X} \times \mathcal{X}$).

Note that the kernel here takes the form $K = \Phi_\infty \Phi_\infty^H$, which for the linear Kernel has the form $K = XX^H$. Furthermore, the kernel matrix must have a similar spectrum to the empirical or sample kernel matrix $K_n = X_n X_n^H$, and indeed for $\lim_{n \to \infty} K_n \to K$, where we use here the definition $X_n \in \mathbb{R}^{d \, times \, n}$, i.e., we have n vectors of dimension d and therefore $\mathcal{X} = \mathbb{R}^d$. The function K is said to be the kernel of L_K and several properties of L_K follow from the properties of K. Since we want to understand the condition number of K_n, the sample covariance matrix, we need to study the behaviour of its eigenvalues. For this, we start by studying the eigenvalues of K. First, from Theorem 2 the next corollary follows.

Corollary 3 ([47], Corollary 3). *The sum $\sum_k \lambda_k$ is convergent, and*

$$\sum_{k=1}^{\infty} \lambda_k = \int_X K(x, x) \leq \nu(X) C_K, \tag{3.37}$$

where $C_K = \sup_{x, x' \in X} |K(x, x')|$ is an upper bound on the kernel. Therefore, for all $k \geq 1$, $\lambda_k \leq \left(\frac{\nu(X) C_K}{k} \right)$.

As we see from Corollary 3, the eigenvalue λ_k (or singular value, since K is SPSD) of L_K cannot decrease slower than $O(1/k)$, since for convergent series of real non-negative numbers $\sum_k \alpha_k$, it must holds that α_k must go to zero faster than $1/k$.

Recalling that K is the *infinite* version of the kernel matrix $K_n = X_n X_n^H$ (or generally $K = \Phi\Phi^H$ for arbitrary kernels) we now need to relate the finite sized Kernel K_n to the kernel K. Leveraging on concentration inequalities for random matrices, we will now show how $K_n \in \mathbb{R}^{d\times d}$ for $X_n \in \mathbb{R}^{d\times n}$ converges to K as n grows, and therefore the spectra (i.e., eigenvalues must match), which implies that the decay of the eigenvalues of K_n. Indeed, we will see that the smallest eigenvalue $\lambda_n = O(1/n)$ with high probability. From this we obtain immediately upper bounds on the condition number in high probability. This is summarised below.

Theorem 4. *The condition number of a Mercer kernel K for a finite number of samples n is with high probability lower bounded by $\Omega(\sqrt{n})$.*

Proof. We will in the following need some auxiliary results.

Theorem 5 (Matrix Bernstein [61])**.** *Consider a finite sequence $\{X_k\}$ of independent, centered, random, Hermitian d-dimensional matrices, and assume that $\mathbb{E}X_k = 0$, and $\|X_k\| \leq R$, for all $k \in [n]$. Let $X := \sum_k X_k$ and $\mathbb{E}X = \sum_k \mathbb{E}X_k$, and let*

$$\sigma(X) = \max\left\{\left\|\mathbb{E}[XX^H]\right\|, \left\|\mathbb{E}[X^HX]\right\|\right\}$$

$$= \max\left\{\left\|\sum_{k=1}^n \mathbb{E}[X_k X_k^H]\right\|, \left\|\sum_{k=1}^n \mathbb{E}[X_k^H X_k]\right\|\right\}. \tag{3.38}$$

Then,

$$\Pr\left[\|X\| \geq \varepsilon\right] \leq 2d\exp\left(\frac{-\varepsilon^2/2}{\sigma(X) + R\varepsilon/3}\right), \forall \varepsilon \geq 0, \tag{3.39}$$

We can make use of this result to straightforwardly bound the largest eigenvalue of the sample covariance matrix, which is a well known result in the random matrix literature. As outlined above, the sample covariance matrix is given by

$$K_n = \frac{1}{n}\sum_{k=1}^n x_k x_k^H, \tag{3.40}$$

for n centred (zero mean) samples in $\mathbb{R}^d$, and $K := \mathbb{E}\left[xx^H\right] \in \mathbb{R}^{d \times d}$. We look in the following at the matrix, i.e., $A := K_n - K$, and assume

$$\|x_i\|_2^2 \leq r, \quad \forall i \in [n] \tag{3.41}$$

i.e., the sample norm is bounded by some constant r. Typically data is sparse, and hence independent of the dimension, although both scenarios are possible. Here we assume a dependency on d. Under this assumption, we let

$$A_k := \frac{1}{n}\left(x_k x_k^H - K\right), \tag{3.42}$$

for each k and hence $A = \sum_{k=1}^n A_k$. With the assumption in Eq. 3.41 we obtain then

$$\|K\| = \left\|\mathbb{E}\left[xx^H\right]\right\| \leq \mathbb{E}\left[\left\|xx^H\right\|\right] = \mathbb{E}\left[\|x\|^2\right] \leq r, \tag{3.43}$$

using Jensen's inequality. The matrix variance statistic $\sigma(A)$ is therefore given by

$$0 \leq \sigma(A) \leq \frac{R}{n}\|C_\infty\| \leq \frac{r^2}{n}, \tag{3.44}$$

which follows from straight calculation. Taking into account that $\|A_k\| \leq \frac{r}{n}$, and by invoking Thm. 5, and using the above bounds. Assuming $r = C \cdot d$ for some constant C, i.e., the norm of the vector will be dependent on the data dimension d (which might not be the case for sparse data!), we hence obtain that

$$\mathbb{E}\left[\|A\|\right] = \mathbb{E}\left[\|K_n - K\|\right] \leq O\left(\frac{d}{\sqrt{n}} + \frac{d}{n}\right) = O\left(\frac{d}{\sqrt{n}}\right). \tag{3.45}$$

Note that this is essentially $O(\sqrt{1/n})$ for n the number of samples, assuming $n \gg d$, and similarly for sparse data with sparsity $s = O(1)$ the norm will not be proportional to d.

We next need to relate this to the eigenvalues λ_{min} and λ_{max}, for which we will need the following lemma.

Lemma 3. *For any two bounded functions f, g, it holds that*

$$\left|\inf_{x \in X} f(x) - \inf_{x \in X} g(x)\right| \leq \sup_{x \in X} |f(x) - g(x)| \tag{3.46}$$

Proof. First we show that $|\sup_X f - \sup_X g| \leq \sup_X |f - g|$. For this, take

$$\sup_X (f \pm g) \leq \sup_X (f - g) + \sup_X g \leq \sup_X |f - g| + \sup_X g,$$

and

$$\sup_X (g \pm f) \leq \sup_X (g - f) + \sup_X f \leq \sup_X |g - g| + \sup_X f,$$

and the result follows. Next we can proof the Lemma by replacing $f = -f$ and $g = -g$ and using that $\inf_X f = \sup_X (-f)$, the claim follows. □

Using that with high probability $\|K_n - K\| \leq O(d/\sqrt{n} + d/n)$, and therefore

$$|\lambda_{max}(K_n) - \lambda_{max}(K)| \leq O\left(d/\sqrt{n}\right), \quad \text{and} \tag{3.47}$$

$$|\lambda_{min}(K_n) - \lambda_{min}(K)| \leq O\left(d/\sqrt{n}\right), \tag{3.48}$$

which follows from Lemma 3 by taking $f(x) = (x^H K_n x)/x^H x$ and $g(x) = (x^H K x)/x^H x$. Therefore, ignoring the data dimension dependency, and recalling that the Kernel matrices are positive semi-definite, we obtain that with high probability

$$\kappa(K_n) = \frac{\lambda_{max}(K_n)}{\lambda_{min}(K_n)} \geq \frac{\|K_n\|}{\lambda_{min}(K) + O(1/\sqrt{n})} \tag{3.49}$$

$$= \frac{\|K_n\|}{O(1/n) + O(1/\sqrt{n})} = O(\|K_n\| \sqrt{n}), \tag{3.50}$$

where we used the bounds on the smallest eigenvalue in Corollary 3. Therefore, in high probability the condition number of the problem is at least $O(\|K_n\| \sqrt{n})$, and if we assume $\|K_n\| = 1$ this is $O(\sqrt{n})$ for any Kernel method. □

We hence can assume that an additional $\sqrt{n}$ dependency from κ can appear as a plausible case. If we additionally take into account the other sources of errors,

which we discussed before, typical QML algorithm result in runtimes which are significantly worse than the classical counterpart. We hence learn from this analysis, that if we desire practically relevant QML algorithms with provable guarantees, we either need to reduce the condition number dependency (e.g., through preconditioning or filtering), or apply the quantum algorithm only to data sets for which we can guarantee that κ is small enough. Since some quantum algorithms allow for spectral filtering methods, and therefore limit the condition number by κ_{eff}, we will not include the condition number scaling in the following analysis. We leave it open to the reader to apply this scaling to algorithms which exhibit a high κ-dependency. Notably, it would furthermore require a more involved analysis of the error since such a truncation immediately imposes an error, e.g., $\|A - F(A)\|_2^2$ for Hermitian $A = U\Sigma U^H$, and filter F, could result in an error $\|\Sigma - F(\Sigma)\|_2^2$. If the filter in the simplest form cuts of the eigenvalues below σ_{eff} then the final error would be σ_{eff}, and it would then be a matter of the propagation of this error through the algorithm.

3.4 Analysis of supervised QML algorithms

Our analysis is based entirely on the dependency of the statistical guarantees of the estimator on the size of the data set. We leverage on the above discussed impacts on the algorithmic error and the measurement based error as well as the previously derived results in statistical learning theory. In particular, by using that the accuracy parameter of a supervised learning problem scales inverse polynomially with the number of samples which are used for the training of the algorithm, we showed that the errors of quantum algorithms will results in poly(n) scalings. The runtimes for a range of quantum algorithms which we now derive based on these requirements indicate that these algorithms can therefore not achieve exponential speedups over their classical counterparts.

We note that this does not rule out exponential advantages for learning problems where no efficient classical algorithms are known, as there exist learning problems for which quantum algorithms have a superopolynomial advantage [62, 63]. One nice feature of our results is that they are independent of the model of access

to the training data, which means that the results hold even if debated access such as quantum random access memory is used. Finally, we note that our results do not assume any prior knowledge on the function to be learned, which allows us to make statements on virtually every possible learning algorithm, including neural networks. Under stronger assumptions (e.g., more knowledge of the target function) the dependency of the accuracy in terms of samples can be derived.

We summarise the results of our analysis in Table 3.1. We omit the κ dependency which would generally decrease the performance of the quantum algorithms further. Notably, while this is classically not an issue due to preconditioning, no efficient general quantum preconditioning algorithm exists. We note that [11] introduced a SPAI preconditioner, however without providing an efficient quantum implementation for its construction and without any performance analysis. We additionally note that recently [64] proposed a different mechanism for constructing efficient preconditioners, called fast inversion. The main idea is based on the fact that fast inversion of 1-sparse matrices can be done efficiently on a quantum computer. The algorithm works for matrices of the form $A + B$, where $\|B\| = \|A^{-1}\| = \|(A+B)^{-1}\| = O(1)$, and A can be inverted fast. It results in a condition number of the QLSA of $\kappa(M(A+B)) = \kappa(I + A^{-1}B)$ once the preconditioner $M = A^{-1}$ is applied.

3.5 Conclusion

Quantum machine learning algorithms promise to be exponentially faster than their classical counter parts. In this chapter, we showed by relying on standard results from statistical learning theory that such claims are not well founded, and thereby rule out QML algorithms with polylogarithmic time complexity in the input dimensions. As any practically-used machine learning algorithms have polynomial runtimes, our results effectively rule out the possibility of exponential advantages for supervised quantum machine learning. Although this holds for polynomial runtime classical algorithms, we note that our analysis does not rule out an exponential advantage over classical algorithms with superpolynomial runtime. Furthermore,

	Algorithm	Train time	Test time
Classical	LS-SVM / KRR	n^3	n
	KRR [?, 65, 66, 67, 68]	n^2	n
	Divide and conquer [51]	n^2	n
	Nyström [54, 55]	n^2	$\sqrt{n}$
	FALKON [10]	$n\sqrt{n}$	$\sqrt{n}$
Quantum	QKLS / QKLR [37]	$\sqrt{n}$	$n\sqrt{n}$
	QSVM [14]	$n\sqrt{n}$	$n^2\sqrt{n}$

Figure 3.1: Summary of time complexities for training and testing of different classical and quantum algorithms when statistical guarantees are taken into account. We omit polylog(n,d) dependencies for the quantum algorithms. We assume $\varepsilon = \Theta(1/\sqrt{n})$ and count the effects of measurement errors. The acronyms in the table refer to: least square support vector machines (LS-SVM), kernel ridge regression (KRR), quantum kernel least squares (QKLS), quantum kernel linear regression (QKLR), and quantum support vector machines (QSVM). Note that for quantum algorithms the state obtained after training cannot be maintained or copied and the algorithm must be retrained after each test round. This brings a factor proportional to the train time in the test time of quantum algorithms. Because the condition number may also depend on n and for quantum algorithms this dependency may be worse, the overall scaling of the quantum algorithms may be slower than the classical.

since we do not make any assumptions on the hypothesis space $\mathscr{H}$ of the learning problem, we note that generally faster error rates are possible if more prior knowledge exists. It is hence possible to obtain faster convergence rates than $1/\sqrt{n}$, which would imply a potential quantum advantage for such problems.

As future directions, it is worth mentioning that it may be possible strengthen our results by analysing the n dependency of the condition number. Previous results in this direction are discussed in [47, 60].

Chapter 4

randomised Numerical Linear Algebra

The research question we answer in this chapter is the following:

Research Question 2 (QML under the lens of RandNLA)**.** *Under the assumption of efficient sampling processes for the data for both classical and quantum algorithms, what is the comparative advantage of the latter?*

We will show how the requirement of a fast memory model can hide much of the computational power of an algorithm. In particular, by allowing a classical algorithm to sample according to a certain probability distribution, we can derive classical algorithms with computational complexities which are independent of the input dimensions, and therefore only polynomially slower compared to the best known quantum algorithm for the same task. We use such Monte Carlo algorithms to construct fast algorithms for Hamiltonian simulation, and also connect our research to the recent so-called *quantum-inspired* or *dequatisation* results [17].

We start by defining a range of memory models, which are used in quantum algorithms, randomised numerical linear algebra, and quantum-inspired algorithms. We continue by introducing the main ideas of Monte Carlo methods for numerical linear algebra, and as an exemplary case study the randomised matrix multiplication algorithm to grasp the main ideas of such approximate methods. For a detailed introduction and overview, we refer the reader to the review of David Woodruff [69].

We then show that such a memory structure immediately leads to faster simulation algorithms for dense Hamiltonians in Sec. 4.4.4.

In the next step, we then use these ideas in Sec. 4.4.5 to construct symmetric matrices which are approximations for the Hamiltonian and use these to perform fast classical Hamiltonian simulation. We do so by first finding a randomised low-rank approximation H_k of the Hamiltonian H, and then applying a form of approximation to the series expansion of the time evolution operator $\exp(-iH_k t)$.

We thereby show that the ability to sample efficiently from such matrices immediately allows for faster classical algorithms as well.

Finally we briefly discuss how our results relate to the recent stream of *dequantisation* results, and show that indeed we can achieve exponentially faster algorithms if we make similar assumptions.

4.1 Introduction

In general, there exist two different approaches to performing approximate numerical linear algebra operations, and a closer inspection shows that there are many parallels between them. The first stream is based on random sampling, also called sub-sampling, while the second stream is based on so-called random projections. Random projections are themselves based on the John-Lindenstrauss transformation, which allow us to embed vectors in a lower dimensional subspace while preserving certain distance metrics. Roughly speaking, random projections correspond to uniform sampling in randomly rotated spaces. The main differences between different approaches are the resulting error bounds, where the best known ones are multiplicative relative-error bounds.

Taking as example the problem of a randomised low-rank approximation, based on the randomised projector $\hat{P}_k$, which projects into some rank-k subspace of a matrix, and denoting with P_k the projector into the subspace with containing the best rank-k approximation of A (e.g., the left eigenspace of the top k singular values of A), then we define additive-error bounds to be of the form

$$\left\| A - \hat{P}_k A \right\|_F \leq \left\| A - P_k A \right\|_F + \varepsilon \left\| A \right\|_F,$$

since we have an additional error term to the best rank-*k* approximation of *A*. These bounds are typically weaker than the so-called *multiplicative-error bounds*, which take the form

$$\left\| A - \hat{P}_k A \right\| \leq f(m,n,k,\dots) \left\| A - P_k A \right\|,$$

where $f(\cdot)$ denotes a function depending on the dimensions *m* and *n* of the matrix $A \in \mathbb{R}^{m \times n}$, the rank, or other parameters. The best known bounds on *f* are independent of *m* or *n*, and are referred to as *constant-factor bounds*. The strongest of these bounds are given for $f = 1 + \varepsilon$, for some error parameter ε, i.e.,

$$\left\| A - \hat{P}_k A \right\| \leq (1 + \varepsilon) \left\| A - P_k A \right\|.$$

4.2 Memory models and memory access

In computer science, many different models for data access and storage are used. Here we briefly recapitulate a number of memory models, which are common in randomised numerical linear algebra, quantum algorithms, and quantum inspired (or dequantised) algorithms. We will introduce the different memory structures, and define their properties. Next, we will introduce the basic ideas behind numerical linear algebra on the example of matrix multiplication.

4.2.1 The pass efficient model

Traditionally, in randomised numerical linear algebra, the so-called pass-efficient model was used to describe memory access. In this model, the only access an algorithm has to the data is via a so-called pass, which is a sequential read of the entire input data set. An algorithm is then called pass-efficient if it uses only a small or a constant number of passes over the data, while it can additionally use RAM space and additional computation sublinear with respect to the data stream to compute the solution.

The data storage can take several forms, as for example one could store only the index-data pairs as a sparse-unordered representations of the data, i.e., $((i,j),A_{ij})$ for all non-zero entries of the matrix *A*. In practice, these types of storage are

for example implemented in the Intel MKL compressed sparse row (CSR) format, which is specified by four arrays (values, columns, pointer 1, and pointer 2).

Stronger results, however, can sometimes be obtained in different input models, yet these memory models are generally inaccessible and hence should be used with care.

In order to efficiently sample from the data set using this memory model, we need to be able to select random samples in a pass-efficient manner. In order to do so, we can rely on the so-called SELECT algorithm, which is presented below.

Algorithm 1 The SELECT Algorithm [70]

INPUT: $\{a_1, \ldots, a_n\}$, $a_i \geq 0$, i.e., one sequential read over the data.
OUTPUT: i^*, a_{i^*}.

1: $D = 0$
2: **for** i=1 to n **do**
3:    $D = D + a_i$
4:    With probability a_i/D, let $i^* = i$ and let $a_{i^*} = a_i$.
5: **end for**
6: **return** i^*, a_{i^*}.

The SELECT algorithm has then the following properties.

Lemma 4 ([71]). *Suppose that* $\{a_1, \ldots, a_n\}, a_i \geq 0$, *are read in one pass, then the SELECT algorithm returns the index i^* with probability* $\mathbb{P}[i^* = i] = a_i / \sum_{j=1}^{n} a_j$, *and requires* $O(1)$ *additional storage space.*

Proof. The proof is by induction. For the base case we have that the first element a_1 with $i^* = 1$ is selected with probability $a_1/a_1 = 1$. The induction step is then performed by letting $D_k = \sum_{j=1}^{k} a_j$, i.e., the first k elements have been read, and the algorithm reads the element $k + 1$. Hence the probability to have selected any prior $i^* = i$ is $\mathbb{P}[i^* = i] = a_i/D_k$. Then the algorithm selects the index $i^* = k+1$ with probability a_{k+1}/D_{k+1}, and retains the previous selection otherwise, which is done for $i \in [k]$ with probability $\mathbb{P}[i^* = i] = \mathbb{P}(\text{Entry was selected prior}) * \mathbb{P}(\text{Current entry was not selected})$. Since by the induction hypothesis it holds that the any entry prior was selected with $\mathbb{P}[i^* = i] = \frac{a_i}{D_k}$ and the probability that the new

entry was not selected is $\left(1 - \frac{a_{k+1}}{D_{k+1}}\right)$, hence we have $\mathbb{P}[i^* = i] = \frac{a_i}{D_k}\left(1 - \frac{a_{k+1}}{D_{k+1}}\right) = \frac{a_i}{D_{k+1}}$. By induction this result holds for all i and hence $l + 1 = n$. The storage space is limited to the space for keeping track of the sum, which is $O(1)$, and hence concludes the proof. $\qquad\square$

It is important to note that we can therefore use the SELECT algorithm to perform importance sampling according to distributions over the rows A_i or columns A^i of a matrix A, as for example if we want to sample according to the probabilities

$$\mathbb{P}[i^* = i] = \|A_i\|_2^2 / \|A\|_F^2 .$$

We will use this sampling scheme to demonstrate how we can already obtain very fast algorithms in randomised linear algebra. In particular, we will use it to derive one of the most fundamental randomised algorithms in Section 4.3: The basic matrix multiplication algorithm.

4.2.2 Quantum random access memory

A classical random-access memory (RAM) is a device that stores the content of a memory location in a memory array. A random-access memory importantly allows data items to be read or written in almost the same amount of time irrespective of the physical location of data inside the memory. Practically this is due to a binary tree which allows the bus to traverse the memory consisting of N elements in $\log(N)$ computational steps by traversing the tree. Quantum random access memory (qRAM) in a similar fashion allows us to access and load data in superposition from all the memory sites. qRAMs with n-bit addresses can therefore access 2^n memory sites and hence require $O(2^n)$ two-bit logic gates. Note that we, with abuse of previous notation, define therefore the dimensions of a Hermitian matrix in this chapter with N, unlike in previous chapters. We do this to comply with the standard notation in the quantum computing literature. Using, e.g. the so-called *bucket brigade* architecture the number of two-qubit physical interactions during each qRAM call can then in principle reduced to $O(n)$ [6, 7], which is hence polylogarithmic in the input dimension, assuming a data array of size $N = 2^n$.

A variety of different qRAM architectures have been proposed, and we will here mainly focus on the specific architecture introduced by Kerenidis and Prakash [16]. We will use this algorithm later on in context of Hamiltonian simulation for dense Hamiltonians on a quantum computer. In the next subsection, we will also show that such a memory model has a powerful classical correspondence which can be used to construct much faster randomised algorithms for numerical linear algebra. In the following, we introduce an adaptation of the architecture proposed in [16], which is suitable for the application that we will mainly study in this chapter, namely Hamiltonian simulation. Notably, while most quantum memory structures store the squares of the entries (e.g., of the matrix or data in general), the here presented one stores the absolute values only.

Definition 2 (Quantum Data Structure). *Let $H \in \mathbb{C}^{N \times N}$ be a Hermitian matrix (where $N = 2^n$), $\|H\|_1$ being the maximum absolute row-sum norm, and $\sigma_j := \sum_k |H_{jk}|$. Each entry H_{jk} is represented with b bits of precision. Define D as an array of N binary trees D_j for $j \in \{0, \ldots, N-1\}$. Each D_j corresponds to the row H_j, and its organization is specified by the following rules.*

1. *The leaf node k of the tree D_j stores the value[1] H_{jk}^* corresponding to the index-entry pair (j, k, H_{jk}).*

2. *For the level immediately above the bottom level, i.e., the leaves, and any node level above the leaves, the data stored is determined as follows: suppose the node has two children storing data a, and b respectively (note that a and b are complex numbers). Then the entry that is stored in this node is given by $(|a| + |b|)$.*

We show an example of the above data structure in Fig. 4.1. As we immediately see from the Definition 2, while each binary tree D_j in the data structure contains a real number in each internal (non-leaf) node, the leaf nodes store a complex number. The root node of each tree D_j then store the absolute column norm, i.e., the value $\sum_{k=0}^{N-1} |H_{jk}^*|$, and we can therefore calculate the value $\|H\|_1 - \sigma_j$ in constant time.

---

[1]Note that the conjugation here is necessary. See Eq. (4.26).

Furthermore, the $\|H\|_1$ can be obtained as the maximum among all the roots of the binary trees, which can be done during the construction of the data structure, or through another binary search through the tree structure which ends in the D_j's.
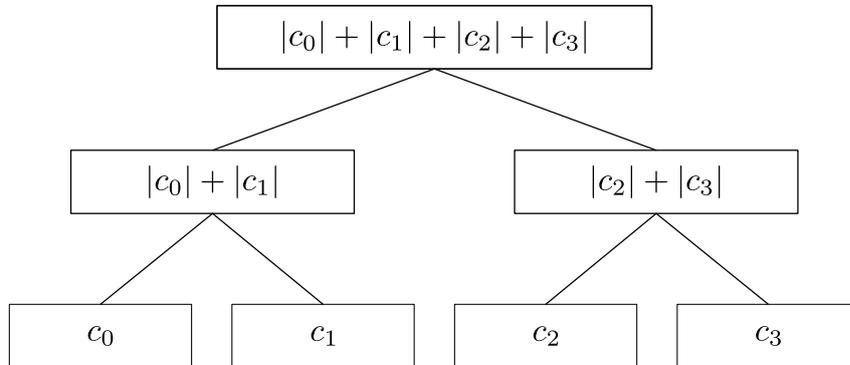
Figure 4.1: An example of the data structure that allows for efficient state preparation using a logarithmic number of conditional rotations.

With this data structure, we can efficiently perform a state preparation which we will require in the Hamiltonian simulation algorithm in Sec. 4.4.4. Notably, as we will see in the next subsection, such a fast quantum memory has a classical equivalent which allows us to sample very efficiently from the data.

There are several challenges with such a memory architecture, including a large overhead from the controlled operations, and possible challenges from decoherence. For a discussion of some caveats of such qRAM architectures, we point the reader to the reviews [72] or [5]. An important notion that we want to mention here is the separation between quantum algorithms which use quantum data (i.e. data which is accessible in superposition) and ones which are operating on classical data which needs to be accessed efficiently to allow polylogarithmic runtimes. In general most quantum algorithms are discussed in the so-called query model, i.e., the computational complexity is given in terms of numbers of queries of an oracle. The actual oracle could then be another quantum algorithm, or a memory model such as qRAM. The computational time is then given by the query complexity times the time for each query call, which could for example be $T_{query} \times \log(N)$ if we call the qRAM T_{query} times. If a quantum algorithm requires qRAM or a similar solution, then we need to understand the associated limitations and assumptions

and assess these with care. Particularly, the superiority w.r.t. classical algorithms is here in question when comparing to classical PRAM machines [73]. Results that consider error probabilities in certain quantum RAM architectures further indicate that a fully-error-corrected quantum RAM is necessary to maintain speedups, which might result in huge overheads and questionable advantages [74, 75].

4.2.3 Quantum inspired memory structures

Recent quantum inspired algorithms rely on a data structure which is very similar to the qRAM used in many QML algorithms. While QML algorithms rely on qRAM or 'quantum access' to data in order to allow for state preparation with linear gate count but polylogarithmic depth, quantum inspired algorithms achieve their polylogarithmic complexities through *sampling and query access* of the data via a *dynamic data structure* [21].

The *Sampling and query access* model can be thought of as a classical analogue of the above introduced qRAM model for state preparation. The ability to prepare a state $|v\rangle$ which is proportional to some input vector $v \in \mathbb{C}^N$ (such as a column of $H \in \mathbb{C}^{N \times M}$) from memory is equivalent to the ability make the following queries:

1. Given an index $i \in [N]$, output the corresponding entry v_i of the vector v.

2. Sample the index $j \in [N]$ with probability $|v_j|^2 / \|v\|_2^2$.

3. Output the spectral norm of the vector $\|v\|_2$.

For a matrix $H \in \mathbb{C}^{M \times N}$ this extends to the ability to perform the following queries:

1. Sample and query access for each vector H_i, i.e., for all rows $i \in [N]$ we can output each entry, sample indices with probability proportional to the magnitude of the entry, and output its spectral norm.

2. Sample and query access to the vector h with $h_i = \|H_i\|$, i.e., the vector of row norms.

Notably, if the input data is given in a classical form, classical algorithms can be run efficiently in the sampling and query model whenever the corresponding quantum algorithms require qRAM access to the data, and both all state preparations

can be performed efficiently (i.e., be performed in logarithmic time in the input dimensions).

The classical data structure which enables such sampling and query access to the data is similar to the one described in Fig. 4.1. It stores a matrix $H \in \mathbb{C}^{M \times N}$, again in form of a set of binary trees, where each tree contains the absolute values (or to be precise, the square of the absolute values) of the entries of one row or column of the input matrix H. The (time) cost of a query to any entry in H is then $O(1)$ and sampling can be performed in time $O(\log(MN))$ for any entry $H \in \mathbb{C}^{M \times N}$. We summarise this data structure below in Definition 3.

Definition 3 (Classical Data Structure). *Let $H \in \mathbb{C}^{N \times M}$ be a Hermitian matrix (where $N = 2^n$), $\|H\|_F$ being the Frobenius norm, and $h_i = \|H_i\|_2$ being the spectral norm of row i of H. Each entry H_{jk} is represented with b bits of precision. Define D as an array of N binary trees D_j for $j \in \{0, \ldots, N-1\}$. Each D_j corresponds to the row H_j, and its organization is specified by the following rules.*

1. *The leaf node k of the tree D_j stores the value $\frac{H_{jk}}{|H_{jk}|}$ corresponding to the index-entry pair (j, k, H_{jk}).*

2. *For the level immediately above the bottom level, i.e., the leaves, the entry k of the tree D_j is given by $|H_{jk}|^2$*

3. *For any node level above the leaves, the data stored is determined as follows: suppose the node has two children storing data a, and b respectively (note that a and b are squares values of complex numbers). Then the entry that is stored in this node is given by $(a + b)$.*

The root nodes of the binary tree D_j is then given by h_j, and the memory structure is completed by applying the same tree structure where now the leaves j of the tree are now given by the root nodes of D_j.

Figure 4.2 demonstrates an example of this structure.

Although we do not explicitly rely on such a data structure for the results of section 4.4.5, since we use a less restrictive requirement which we call row-

$$\|H\|_{\mathrm{F}}$$

$$h_1 = \|H_1\|_2^2 \qquad h_2 = \|H_2\|_2^2$$

$$|H_{11}|^2 + |H_{12}|^2 \quad |H_{13}|^2 + |H_{14}|^2 \qquad |H_{21}|^2 + |H_{22}|^2 \quad |H_{23}|^2 + |H_{24}|^2$$

$$|H_{11}|^2 \quad |H_{12}|^2 \quad |H_{13}|^2 \quad |H_{14}|^2 \qquad |H_{21}|^2 \quad |H_{22}|^2 \quad |H_{23}|^2 \quad |H_{24}|^2$$

$$\frac{H_{11}}{|H_{11}|} \quad \frac{H_{12}}{|H_{12}|} \quad \frac{H_{13}}{|H_{13}|} \quad \frac{H_{14}}{|H_{14}|} \qquad \frac{H_{21}}{|H_{21}|} \quad \frac{H_{22}}{|H_{22}|} \quad \frac{H_{23}}{|H_{23}|} \quad \frac{H_{24}}{|H_{24}|}$$

Figure 4.2: An example of the classical (dynamic) data structure that enables efficient sample and query access for the example of $H \in \mathbb{C}^{2 \times 4}$.

computability and row-searchability. However, the data structure in Def. 3 immediately allows us to perform these operations. We will in this thesis just refer to this requirement as *query and sample access*, since it has been established as the commonly used term in the dequantisation literature and suffices in practice. There are generally exceptions, for example, since row-computability and row-searchability can also be achieved through structural properties of the matrix which would not require the memory structure.

4.3 Basic matrix multiplication

In this section we will introduce some of the main concepts from randomised numerical linear algebra on the example of the randomised matrix multiplication algorithm.

We will denote the i-th column of some matrix A by A^i and the j-th row of A by A_j. Let in the following $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$. With this notation, recall that $(AB)_{ij} = \sum_{j=1}^n A_{ij} B_{jk} = A_i B^j$, and that we can write the product $AB = \sum_{i=1}^n A^i B_i$ via the sum of outer products, i.e., a sum of rank-1 matrices, where $A^i B_i \in \mathbb{R}^{m \times p}$. The representation of the product AB into the sum of outer products implies that AB could be decomposed into a sum of random variables, which, if appropriately chosen, would in expectation result in the product AB. This suggests that we could sample such terms to approximate the product, specifically we could use an approx-

imation of the form

$$AB = \sum_{i=1}^{n} A^i B_i \approx \frac{1}{c} \sum_{t=1}^{c} \frac{A^{i_t} B_{i_t}}{p_{i_t}} = CR, \tag{4.1}$$

where $\{p_i\}_{i=1}^{n}$ are the sampling probabilities. We could do this via random uniform sampling, but this would lead to a very high variance. Hence we will further optimally like to optimise the sampling probabilities to obtain a good result.

In general, instead of working with probabilities, it is easier to work with matrices, and we can define a standardized matrix notation called *sampling matrix formalism* according to [71]. Notably, we can treat many other approaches in the framework of matrix-multiplication, and will therefore also rely on it in the following. For this we let $S \in \mathbb{R}^{n \times c}$ be a matrix such that

$$S_{ij} := \begin{cases} 1 & \text{if the } i\text{-th column of } A \text{ is chosen in the } j\text{-th independent trial} \\ 0 & \text{otherwise,} \end{cases} \tag{4.2}$$

and let $D \in \mathbb{R}^{c \times c}$ be a diagonal matrix such that

$$D_{tt} = 1/\sqrt{c p_{i_t}}. \tag{4.3}$$

Using this, we can write the output of the sampling process described in Eq. 4.1 via the simplified notation

$$CR = ASD(SD)^T B \approx AB. \tag{4.4}$$

The matrix multiplication algorithm is given below in Algorithm 2, and below we will next give a proof of its correctness.

In the following analysis we will use the fact that for rank-1 matrices it holds that $\left\| A^i B_i \right\|_2 = \left\| A^i \right\|_2 \left\| B_i \right\|_2$. In order to see why this is the case observe that

$$\left\| A^i B_i \right\|_2 = \sqrt{(A^i B_i)^T (A^i B_i)} = \sqrt{B_i^T (A^i)^T A^i B_i},$$

Algorithm 2 The BASIC-MATRIX-MULTIPLICATION Algorithm [70]

INPUT: $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, integer $c > 0$ and sampling probabilities $\{p_i\}_{i=1}^n$.
OUTPUT: C and R s.t. $CR \approx AB$.

1: C, R as all-zeros matrices.
2: **for** i=1 to c **do**
3:     Sample an index $i_t \in \{1, \dots, n\}$ w. prob. $\mathbb{P}[i_t = k] = p_k$, i.i.d with replacement.
4:     Set $C^t = A^{i_t}/\sqrt{cp_{i_t}}$ and $R_t = B_{i_t}/\sqrt{cp_{i_t}}$.
5: **end for**
6: **return** C and R.

---

but this is simply the inner product and hence

$$\sqrt{\sigma_{max}(\|B_i\|_2^2 \|A^i\|_2^2)} = \sqrt{\|B_i\|_2^2 \|A^i\|_2^2} = \|B_i\|_2 \|A^i\|_2.$$

We now prove a Lemma that states that *CR* is an unbiased estimator for *AB*, element-wise, and calculate the variance of that estimator. This variance is strongly dependent on the sampling probabilities. Based on this, we can then derive optimal sampling probabilities which minimise the variance. In practice these will not be accessible and one typically needs to find approximations for these. In practice, we therefore use other distributions, such as the row or column norms, or so-called leverage scores [71, 69].

Lemma 5 ([70])**.** *Given two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$, construct matrices C and R with the matrix multiplication algorithm from above. Then,*

$$\mathrm{E}[(CR)_{ij}] = (AB)_{ij}, \tag{4.5}$$

$$\mathrm{Var}[(CR)_{ij}] = \frac{1}{c} \sum_{k=1}^n \frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{c}(AB)_{ij}^2 \tag{4.6}$$

Proof. Fix an index-pair (i, j) and define the random variable $X_t = \left(\frac{A^{i_t} B_{i_t}}{cp_{i_t}}\right)_{ij} = \left(\frac{A_{ii_t} B_{i_t j}}{cp_{i_t}}\right)$, and observe that $(CR)_{ij} = \sum_{t=1}^c X_t$. Then we have that $\mathrm{E}[X_t] = \sum_{k=1}^n p_k \frac{A_{ik} B_{kj}}{cp_k} = \frac{1}{c}(AB)_{ij}$, and $\mathrm{E}[X_t^2] = \sum_{k=1}^n \frac{A_{ik}^2 B_{kj}^2}{c^2 p_k}$. Furthermore we just need to

sum up in order to obtain $E[(CR)_{ij}]$, and hence obtain

$$E[(CR)_{ij}] = \sum_{t=1}^{c} E[X_t] = (AB)_{ij},$$

and $\mathrm{Var}[(CR)_{ij}] = \sum_{t=1}^{c} \mathrm{Var}[X_t]$, where we can determine the variance of X_t easily from $\mathrm{Var}[X_t] = E[X_t^2] - E[X_t]^2$, and the lemma follows. $\square$

We can use this directly to establish the expected error in terms of the Frobenius norm, i.e., $E[\|AB - CR\|_F^2]$, by observing that we can treat this as a sum over each of the individual entries, i.e.,

$$E[\|AB - CR\|_F^2] = \sum_{i=1}^{m} \sum_{j=1}^{p} E\left[(AB - CR)_{ij}^2\right].$$

Doing so, we obtain the following result.

Lemma 6 (Basic Matrix Multiplication [70])**.** *Given matrices A and B, construct matrices C and R with the matrix multiplication algorithm from above. Then it holds that,*

$$E[\|AB - CR\|_F^2] = \sum_{k=1}^{n} \frac{\left\|A^k\right\|_2^2 \left\|B_k\right\|_2^2}{c p_k} - \frac{1}{c} \|AB\|_F^2 \tag{4.7}$$

Furthermore, if

$$p_k = p_k^{optimal} = \frac{\left\|A^k\right\|_2 \left\|B_k\right\|_2}{\sum_{k'} \left\|A^{k'}\right\|_2 \left\|B_{k'}\right\|_2},$$

are the sampling probabilities used, then

$$E[\|AB - CR\|_F^2] = \frac{\left(\sum_{k=1}^{n} \left\|A^k\right\|_2 \left\|B_k\right\|_2\right)^2}{c} - \frac{1}{c} \|AB\|_F^2. \tag{4.8}$$

Proof. As mentioned above, first note that

$$E[\|AB - CR\|_F^2] = \sum_{i=1}^{m} \sum_{j=1}^{p} E\left[(AB - CR)_{ij}^2\right].$$

Oberve that the squared terms give $(AB - CR)_{ij}^2 = (AB)_{ij}^2 - (AB)_{ij}(CR)_{ij} - (CR)_{ij}(AB)_{ij} + (CR)_{ij}^2$, and using the results for $E[(CR)_{ij}]$ from Lemma 5 and

since $\mathrm{E}[(ABCR)_{ij}] = (AB)_{ij}^2$, we have

$$\mathrm{E}[(AB-CR)_{ij}^2] = \mathrm{E}[(CR)_{ij}^2] - (AB)_{ij}^2 = \mathrm{E}[(CR)_{ij}^2] - \mathrm{E}[(CR)_{ij}]^2 = \mathrm{Var}[(CR)_{ij}].$$

Therefore we have $\mathrm{E}[\|AB-CR\|_F^2] = \sum_{i=1}^m \sum_{j=1}^p \mathrm{Var}[(CR)_{ij}]$. Using the result from Lemma 5 again for the variance, we then directly obtain the result by using that $\left(\sum_i A_{ik}^2\right) = \|A^k\|_2^2$ and $\left(\sum_j B_{kj}^2\right) = \|B_k\|_2^2$. $\qquad\square$

The sampling probabilities that we have used here are optimal in the sense that they minimize the expected error. To show this we can define the function

$$f(\{p_i\}_{i=1}^n) = \sum_{k=1}^n \frac{\|A^k\|_2^2 \|B_k\|_2^2}{p_k},$$

which captures the p_k-dependent part of the error. In order to find the optimal probabilities we minimise this with the constraint that $\sum_k p_k = 1$, i.e., we define the function $g = f(\{p_i\}_{i=1}^n) + \lambda\left(\sum_k p_k - 1\right)$. Setting the derivative of this function w.r.t. the p_k to zero and correctly normalising them then gives the probabilities in Lemma 6. Note a few important points about this result.

- Note the results depends strongly on $\{p_k\}_{k=1}^n$. In particular we need a strategy to obtain these probabilities.

- With Markov's inequality we can remove the expectation (using the assumption that $p_k \geq \beta * p_k^{optimal}$). To do this we reformulate Eq. 4.8 and incorporate the second term such that we obtain for some $1 \geq \beta > 0$ and hence nearly optimal probabilities for β sufficiently close to 1,

$$\mathrm{E}[\|AB-CR\|_F^2] = \frac{\left(\sum_{k=1}^n \|A^k\|_2 \|B_k\|_2\right)^2}{\beta c} - \frac{1}{c}\|AB\|_F^2 \leq \frac{1}{\beta c}\|A\|_F^2 \|B\|_F^2, \tag{4.9}$$

where we neglect the last term and used the Cauchy-Schwarz inequality, i.e.,

$$\left(\sum_{k=1}^n \|A^k\|_2 \|B_k\|_2\right)^2 \leq \sum_{k=1}^n \|A^k\|_2^2 \sum_{k=1}^n \|B_k\|_2^2.$$

Now we can apply Jensen's inequality such that $E[\|AB - CR\|_F] \leq \frac{1}{\sqrt{\beta c}} \|A\|_F \|B\|_F$. Hence if we take the number of samples $c \geq 1/(\beta \varepsilon^2)$, then we obtain a bound of the error $E[\|AB - CR\|_F] \leq \varepsilon \|A\|_F \|B\|_F$. We can now use Markov's inequality to *remove the expectation* from this bound, and in some cases this will be good enough. Let

$$\mathbb{P}\left[\|AB - CR\|_F > \frac{\alpha}{\sqrt{\beta c}} \|A\|_F \|B\|_F\right],$$

we obtain using Markov's inequality that

$$\delta \leq \frac{E[\|AB - CR\|_F]}{\frac{\alpha}{\sqrt{\beta c}} \|A\|_F \|B\|_F} \leq \frac{1}{\alpha}.$$

And hence with probability $\geq 1 - \delta$ if $c \geq \beta / \delta^2 \varepsilon^2$ we have that $\|AB - CR\|_F \leq \varepsilon \|A\|_F \|B\|_F$.

While the above bounds are already good, if we desire a small failure probability δ, the number of samples grow too rapidly to be useful in practice. However, the above results can be exponentially improved using much stronger Chernoff-type bounds. These indeed allow us to reduce the number of samples to $O(\log(1/\delta))$, rather than $\text{poly}(1/\delta)$, and therefore lead to much more practically useful bounds. We do not go into much detail here, as we believe the main ideas are covered above and the details are beyond the scope of this thesis. Next, we first use the quantum random access memory from Def. 2 to derive a fast quantum algorithm for Hamiltonian simulation, and then show how randomised numerical linear algebra can be used to design fast classical algorithms for the same task.

4.4 Hamiltonian Simulation

In this section, we will now derive two results. First, we will derive a quantum algorithm for Hamiltonian simulation, which is based on the qRAM model that we described above in Def. 2. Next, we will design a classical version for Hamiltonian simulation which is independent of the dimensionality which depends on the Frobe-

nius norm, and therefore on the spectral norm and the rank of the input Hamiltonian. The classical algorithms relies on the classical data structure 3.

4.4.1 Introduction

Hamiltonian simulation is the problem of simulating the dynamics of quantum systems, i.e., how a quantum system evolves over time. Using quantum computers to describe these dynamics was the original motivation by Feynman for quantum computers [76, 77]. It has been shown that quantum simulation is BQP-hard, and therefore it was conjectured that no classical algorithm can solve it in polynomial time, since such an algorithm would be able to efficiently solve any problem which can be solved efficiently with a quantum algorithm, including integer factorization [78].

One important distinction between the quantum algorithm which we present in this thesis, and quantum algorithms which have been traditionally developed is the input model. A variety of input models have been considered in previous quantum algorithms for simulating Hamiltonian evolution. The *local Hamiltonian* model is defined by a number of local terms of a given Hamiltonian. On the other hand, the *sparse-access* model for a Hamiltonian H with sparsity s, i.e., the number of non-zero entries per row or column, is specified by the following two oracles:

$$O_S \left| i, j \right\rangle \left| z \right\rangle \mapsto \left| i, j \right\rangle \left| z \oplus S_{i,j} \right\rangle, \text{ and} \tag{4.10}$$

$$O_H \left| i, j \right\rangle \left| z \right\rangle \mapsto \left| i, j \right\rangle \left| z \oplus H_{i,j} \right\rangle, \tag{4.11}$$

for $i, j \in [N]$, where $S_{i,j}$ is the j-th nonzero entry of the i-th row of $H \in \mathbb{R}^{N \times N}$ and $\oplus$ denotes the bit-wise XOR. Note that H is a Hermitian matrix, and therefore square. The *linear combination of unitaries* (*LCU*) model decomposes the Hamiltonian into a linear combination of unitary matrices and we are given the coefficients and access to an implementation of each unitary.

The first proposal for an implementation of Hamiltonian simulation on a quantum computer came from Lloyd [8], and was based on local Hamiltonians. Later, Aharonov and Ta-Shma described an efficient algorithm for an arbitrary sparse Hamiltonian [79], which was only dependent on the sparsity instead of the di-

mension N. Subsequently, a wide range of algorithms have been proposed, each improving the runtime [80, 81, 82, 83, 84, 85, 86, 87, 88, 89]. Most of these algorithms have been defined in the sparse-access model, and have lead to optimal dependence on all (or nearly all) parameters for sparse Hamiltonians over the recent years [90, 91, 92].

The above-mentioned input models are highly relevant when we want to simulate a physical system, for example to obtain the ground state energy of a small molecule. However, these models are not generally used in modern machine learning or numerical linear algebra algorithms, as we have seen above. Here it can be more convenient to work with access to a *quantum random access memory* (*qRAM*) model, which we have described in Def. 2. In this model, we assume that the entries of a Hamiltonian are stored in a binary tree data structure [16], and that we have quantum access to the memory. Here, quantum access implies that the qRAM is able to efficiently prepare quantum states corresponding to the input data. The use of the qRAM model has been successfully demonstrated in many applications such as quantum principal component analysis [13], quantum support vector machines [14], and quantum recommendation systems [16], among many other quantum machine learning algorithms.

In contrast to prior work, we consider the qRAM model in order to simulate not necessarily sparse Hamiltonians. The qRAM allows us to efficiently prepare states that encode the rows of the Hamiltonian. Using the combination of this ability to prepare states in combination with a quantum walk [90], we derive the *first* Hamiltonian simulation algorithm in the qRAM model whose time complexity has $\widetilde{O}(\sqrt{N})$ dependence, where $\widetilde{O}(\cdot)$ hides all poly-logarithmic factors, for non-sparse Hamiltonians of dimensionality N. Our results immediately imply [93] a quantum linear system algorithm in the qRAM model with square-root dependence on dimension and poly-logarithmic dependence on precision, which exponentially improves the precision dependence of the quantum linear systems algorithm by [94].

The main hurdle in quantum-walk based Hamiltonian simulation is to efficiently prepare the states which allow for a quantum walk corresponding to

$e^{-iH/\|H\|_1}$. These states are substantially different to those which have been used in previous quantum algorithms [16, 81, 90]. Concretely, the states required in previous algorithms, such as [90] allow for a quantum walk corresponding to $e^{-iH/(s\|H\|_{\max})}$, where s is the row-sparsity of H. These states can be prepared with $O(1)$ queries to the sparse-access oracle. However, the states we are required to prepare for the non-sparse Hamiltonian simulation algorithm cannot use structural features such as sparsity of the Hamiltonian, and it is not known how to prepare such state efficiently in the sparse-access model. In the qRAM model on the other hand, we are able to prepare such states with time complexity (circuit depth) of $O(\text{polylog}(N))$, as we will demonstrate below.

Using the efficient state preparation procedure in combination with a linear combination of quantum walks, we are able to simulate the time evolution for non-sparse Hamiltonians with only poly-logarithmic dependency on the precision. The main result of this chapter is summarised in the following theorem, which we prove in Sec. 4.4.4.

Theorem 6 (Non-sparse Hamiltonian Simulation)**.** *Let $H \in \mathbb{C}^{N \times N}$ (with $N = 2^n$ for a n qubit system) be a Hermitian matrix stored in the data structure as specified in Definition 2. There exists a quantum algorithm for simulating the evolution of H for time t and error ε with time complexity (circuit depth)*

$$O\left(t \|H\|_1 n^2 \log^{5/2}(t \|H\|_1 /\varepsilon)\frac{\log(t \|H\| /\varepsilon)}{\log\log(t \|H\| /\varepsilon)}\right). \tag{4.12}$$

Here $\|\cdot\|_1$ denotes the induced 1-norm (i.e., maximum absolute row-sum norm), defined as $\|H\|_1 = \max_j \sum_{k=0}^{N-1} |H_{jk}|$, $\|\cdot\|$ denotes the spectral norm. In the following we will also need the max norm $\|\cdot\|_{\max}$, defined by $\|H\|_{\max} = \max_{i,j} |H_{jk}|$.

Since it holds that $\|H\|_1 \leq \sqrt{N} \|H\|$ (see [95]), we immediately obtain the following corollary.

Corollary 7. *Let $H \in \mathbb{C}^{N \times N}$ (where $N = 2^n$) be a Hermitian matrix stored in the data structure as specified in Definition 2. There exists a quantum algorithm for simulating the evolution of H for time t and error ε with time complexity (circuit*

depth)

$$O\left(t\sqrt{N}\,\|H\|\,n^2\log^{5/2}(t\sqrt{N}\,\|H\|/\varepsilon)\frac{\log(t\,\|H\|/\varepsilon)}{\log\log(t\,\|H\|/\varepsilon)}\right). \qquad (4.13)$$

Remarks:

1. As we can see from Corollary 7, the circuit depth scales as $\widetilde{O}(\sqrt{N})$. However, the gate complexity can in principle scale as $O(N^{2.5}\log^2(N))$ due to the addressing scheme of the qRAM, as defined in Definition 2. For structured Hamiltonians H, the addressing scheme can of course be implemented more efficiently.

2. If the Hamiltonian H is s-sparse, i.e., H has at most s non-zero entries in each row or column, then the time complexity (circuit depth) of the algorithm is given by

$$O\left(t\sqrt{s}\,\|H\|\,n^2\log^{5/2}(t\sqrt{s}\,\|H\|/\varepsilon)\frac{\log(t\,\|H\|/\varepsilon)}{\log\log(t\,\|H\|/\varepsilon)}\right). \qquad (4.14)$$

To show this, we need the following proposition.

Proposition 1. *If $H \in \mathbb{C}^{N\times N}$ has at most s non-zero entries in any row, it holds that $\|A\|_1 \leq \sqrt{s}\,\|A\|$.*

Proof. First observe that $\|A\|^2 \leq \sum_i \lambda_i(A^H A) = \text{Tr}\left(A^H A\right) = \|A\|_F^2$, and furthermore we have that $\sum_{ij}|a_{ij}|^2 \leq s\max_{j\in[N]}\sum_i|a_{ij}|^2$. From this we have that $\|A\| \leq \sqrt{s}\,\|A\|_1$ for a s-sparse A. By [96, Theorem 5.6.18], we have that $\|A\|_1 \leq C_M(1,*)\,\|A\|$ for $C_M(1,*) = \max_{A\neq 0}\frac{\|A\|}{\|A\|_1}$ and using the above we have $C_M(1,*) \leq \sqrt{s}$. Therefore we find that $\|A\|_1 \leq \sqrt{s}\,\|A\|$ as desired. $\qquad\square$

The result then immediately follows since $\|A\|_1 \leq \sqrt{N}\,\|A\|$ for dense A, i.e., using $\|H\|_1 \leq \sqrt{s}\,\|H\|$ from Proposition 1 with $s = N$, and the result from Theorem 6.

3. We note that we could instead of qRAM also rely on the sparse-access model in order to prepare the states given in Eq. (4.17). The time complexity of the

state preparation for the states in Eq. (4.17) in the sparse-access model results in an additional $O(s)$ factor (for computing σ_j) compared to the qRAM model, and therefore in a polynomial slowdown. Therefore, in order to simulate H for time t in the sparse-access model, the time complexity of the algorithm in terms of t, d, and $\|H\|$ results in $O(ts^{1.5}\|H\|)$ as $\|H\|_1 \leq \sqrt{s}\|H\|$, which implies that the here presented methods do not give any advantage over previous results in the sparse-access model.

Similarly to the quantum algorithm, we have also designed a fast classical algorithm for Hamiltonian simulation which indeed also only depends on the sparsity of the matrix and the norm. Our classical algorithm requires H to be row-searchable and row-computabile, which we will define later. In short, these requirements are both fulfilled if we are given access to the memory structure described in Definition 3. Informally our results can be summarised as follows:

Theorem 8 (Hamiltonian Simulation With The Nyström Method (Informal version of Theorem 11))**.** *Let $H \in \mathbb{C}^{N \times N}$ be a Hermitian matrix, which is stored in the memory structure given in Definition 3 and at most s non-zero entries per row, and if $\psi \in \mathbb{C}^N$ is an n-qubit quantum state with at most q entries. Then, there exists an algorithm that, with probability of at least $1 - \delta$, approximates any chosen amplitude of the state $e^{iHt}\psi$ in time*

$$O\left(sq + \frac{t^9 \|H\|_F^4 \|H\|^7}{\varepsilon^4}\left(n + \log\frac{1}{\delta}\right)^2\right),$$

up to error ε in spectral norm, where $\|\cdot\|_F$ is the Frobenius norm, $\|\cdot\|$ is the spectral norm, s is the maximum number of non-zero elements in the rows or columns of H, and q is the number of non-zero elements in ψ.

Our algorithm is efficient in the low-rank and sparse regime, so if we compare our algorithm to the best quantum algorithm in the sparse-input model or the black-box Hamiltonian simulation model, only a polynomial slowdown occurs. A result which was recently published [21] is efficient when H is only low-rank and in comparison their time complexity scales as $\text{poly}(t, \|H\|_F, 1/\varepsilon)$, where $\|H\|_F$ is

the Frobenius norm of H. Notably, our algorithm has therefore stricter requirements, such as the sparsity of the Hamiltonian and the sparsity of the input state $|\psi\rangle$, however we achieve much lower polynomial dependencies.

By analysing the dependency of the runtime on the Frobenius norm we can determine under which conditions we can obtain efficient (polylogarithmic runtime in the dimensionality). Hamiltonian simulations. Informally, we obtain:

Corollary 9 (Informal)**.** *If H is a Hamiltonian on n qubits with at most $s = O(\text{polylog}(N))$ entries per row such that $\|H\|_F^2 - \frac{1}{N}\text{Tr}(H)^2 \leq O(\text{polylog}(N))$, and if ψ is an n-qubit quantum state with at most $q = O(\text{polylog}(N))$ entries, then there exists an efficient algorithm that approximates any chosen amplitude of the state $e^{iHt}\psi$.*

While these results have rule out the possibility of exponential speedups of our quantum algorithm in the low-rank regime, we note that the complexity of the quantum algorithm has a much lower degree in the polynomials compared to the classical algorithms. The quantum algorithm has hence still a large polynomial speedup over the classical algorithm (for low-rank Hamiltonians and dense Hamiltonians).

4.4.2 Related work

As already discussed earlier, there are a range of previous results. Many of these are given in different data access models so we will in the following briefly discuss the main results in the respective access model.

Hamiltonian simulation with $\|H\|$ dependence. The black-box model for Hamiltonian simulation is a special case of the sparse-access model. It is suitable for non-sparse Hamiltonians and allows the algorithm to query the oracle with an index-pair $|i,j\rangle$, which then returns the corresponding entry of the Hamiltonian H, i.e., O_H defined in Eq. (4.11). Given access to a Hamiltonian in the black-box model allows to simulate H with error ε with query complexity

$$O((\|H\|t)^{3/2}N^{3/4}/\sqrt{\varepsilon})$$

for dense Hamiltonians [81]. Empirically, the authors of [81] additionally observed, that for several classes of Hamiltonians, the actual number of queries required for simulation even reduces to

$$O(\sqrt{N}\log(N)).$$

However, this $\widetilde{O}(\sqrt{N})$ dependence does not hold provably for all Hamiltonians, and was therefore left as an open problem. After the first version of this work was made public, this open problem was almost resolved by Low [92], who proposed a quantum algorithm for simulating black-box Hamiltonians with time complexity

$$O((t\sqrt{N}\|H\|)^{1+o(1)}/\varepsilon^{o(1)}).$$

We note that the qRAM model is stronger than the black-box model, and therefore the improvements could indeed stem from the input model. However, our model to date gives the best performance for quantum machine learning applications, and therefore might overall allow us to achieve superior performance in terms of computational complexity.

Hamiltonian simulation with $\|H\|_{\max}$ dependence. As previously noted, the qRAM model is stronger than the black-box model and the sparse-access model. One immediate implication of this is that prior quantum algorithms such as [81, 90] can immediately be used to simulate Hamiltonians in the qRAM model. Using black-box Hamiltonian simulation for a s-sparse Hamiltonian, the circuit depth is then given by $\widetilde{O}(ts\|H\|_{\max})$ [81, 90]. For non-sparse H, this implies a scaling of $\widetilde{O}(tN\|H\|_{\max})$. Since $\|H\| \leq \sqrt{N}\|H\|_{\max}$ implies $\widetilde{O}(t\sqrt{H}\|H\|) = \widetilde{O}(tN\|H\|_{\max})$, our results do not give an advantage if the computational complexity is expressed in form of the $\|H\|_{\max}$. However, we can express the computational complexity in terms of the $\|H\|$, which plays a crucial role in solving linear systems [36, 93], and black-box unitary implementation [81]. In this setting, we achieve a quadratic improvement w.r.t. the dimensionality dependence, as the inequality $\|H\|_{\max} \leq \|H\|$ implies $\widetilde{O}(tN\|H\|_{\max}) = \widetilde{O}(tN\|H\|)$.

Hamiltonian simulation in the qRAM model. Shortly after the first version of our results were made available, Chakraborty, Gilyén, and Jeffery [37] independently proposed a quantum algorithm for simulating non-sparse Hamiltonians. Their algorithm makes use of a qRAM input model similar to the one proposed by Kerenidis [16], and achieved the same computational complexity as our result. However, their work is also generalising our results since it uses a more general input model, namely, the block-encoding model in which it first frames the results. This was first proposed in [87], and it assumes that we are given a unitary $\left(\begin{smallmatrix} H/\alpha & \cdot \\ \cdot & \cdot \end{smallmatrix} \right)$ that contains the Hamiltonian H/α (where we want to simulate H) in its upper-left block. The time evolution e^{-iHt} is then performed in $\widetilde{O}(\alpha \|H\| t)$ time. Using the sparse-access model for s-sparse Hamiltonian H, a block-encoding of H with $\alpha = d$ can be efficiently implemented, implying an algorithm for Hamiltonian simulation with time complexity [87]

$$\widetilde{O}(s \|H\| t).$$

The main result for Hamiltonian simulation described in [37] then takes the qRAM model for a s-sparse Hamiltonian H, and shows that a block-encoding with $\alpha = \sqrt{s}$ can be efficiently implemented. This yields an algorithm for Hamiltonian simulation with time complexity

$$\widetilde{O}(\sqrt{s} \|H\| t).$$

We note that the techniques introduced in [37] have been generalised in [97] to a quantum framework for implementing singular value transformation of matrices. Furthermore, [37] gives a detailed analysis which applies their results to the quantum linear systems algorithm, which we have omitted in our analysis.

Quantum-inspired classical algorithms for Hamiltonian simulation. The problem of applying functions of matrices has been studied intensively in numerical linear algebra. The exponential function is one application which has received particular interest in the literature [98, 99, 100, 101]. For arbitrary Hermitian matrices, at this point in time, no known algorithm exists that exhibits a runtime logarithmic in the dimension of this input matrix. However, such runtimes are required if we

truly want to simulate the time evolution of quantum systems, as the dimensions of the matrix that governs the evolution scale exponentially with the number of quantum objects (such as atoms or orbitals) in the system.

More recently, the field of randomised numerical linear algebra techniques, from which we have previously seen the example of randomised matrix multiplication (c.f. Section 4.3) has enabled new approaches to such problems. These methods, along with results from spectral graph theory, have culminated in a range of new classical algorithms for matrix functions. In particular, they have also recently given new algorithms for approximate matrix exponentials [70, 102, **?**, 69, 55]. Previous results typically hold for matrices, which offer some form of structure. For example, Orrecchia *et al.* [103] combined function approximations with the spectral sparsifiers of Spielmann and Teng [104, 105] into a new algorithm that can approximate exponentials of strictly diagonally dominant matrices in time almost linear in the number of non-zero entries of H. A standard approach is to calculate low-rank approximations of matrices, and then use these within function-approximations to implement fast algorithms for matrix functions [106]. Although these methods have many practical applications, they are not suitable to the application at hand, i.e., Hamiltonian simulation, since they produce sketches that do not generally preserve the given symmetries of the input matrix.

For problems where the symmetry of the sketched matrix is preserved, alternative methods such as the Nyström method have been proposed. The Nyström method has originally been developed for the approximation of kernel matrices in statistical learning theory. Informally, Nyströmmethods construct a lower-dimensional, symmetric, positive semidefinite approximation of the input matrix by sampling from the input columns. More specifically, let $A \in \mathbb{R}^{N \times N}$ be a symmetric, rank r, positive semidefinite matrix, A^j the j-th column vector of A, and A_i the i-th row vector of A, with singular value decomposition $A = U\Sigma U^H$ ($\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r)$), and Moore-Penrose pseudoinverse $A^+ = \sum_{t=1}^{r} \sigma_t^{-1} U^i (U^i)^H$. The Nyström method then finds a low-rank approximation for the input matrix A which is close to A in spectral norm (or Frobenius norm), which also preserves the

symmetry and positive semi-definiteness property of the matrix. Let C denote the $n \times l$ matrix formed by (uniformly) sampling $l \ll n$ columns of A, W denote the $l \times l$ matrix consisting of the intersection of these l columns with the corresponding l rows of A, and W_k denote the best rank-k approximation of W, i.e.,

$$W_k = \mathrm{argmin}_{V \in \mathbb{R}^{l \times l}, \mathrm{rank}(V)=k} \|V - W\|_F .$$

The Nyström method therefore returns a rank-k approximation $\tilde{A}_k$ of A for $k < n$ defined by:

$$\tilde{A}_k = CW_k^+ C^\top \approx A$$

The running time of the algorithm is $O(nkl)$ [107]. There exist many ways of sampling from the initial matrix A in order to create the approximation $\tilde{A}_k$, and in particular non-uniform sampling schemes enable us to improve the performance, see for example Theorem 3 in [108]. The Nyström method results in particularly good approximations for matrices which are approximately low rank. The first applications it was developed for were regression and classification problems based on Gaussian processes, for which Williams and Seeger developed a sampling-based algorithm [54, 54, 109]. Since the technique exhibit similarities to a method for solving linear integral equations which was developed by Nyström [110], they denoted their result as *Nyström method*. Other methods, which have been developed more recently include the *Nyström extension*, which has found application in large-scale machine learning problems, statistics, and signal processing [54, 109, 111, 112, 113, 107, 114, 115, 116, 117, 118, 111, 119].

After the here presented quantum algorithm and our Nyström algorithm for Hamiltonian simulation which can use the memory model from Def. 3 have been made public, a general framework was proposed to perform such quantum-inspired classical algorithms which are based on the classical memory structure from Def. 3 [21], namely *sampling and query access*. Notably, this memory structure was first proposed by Tang [17]. Although their framework generalises all quantum-inspired algorithm using the input model from Def. 3, our classical al-

gorithm does not require the specified input model. Furthermore our algorithm achieves significantly lower polynomial dependencies. While the time complexity of their algorithm has a 36-th power dependency on the Frobenius norm of the Hamiltonian, our algorithm scales with a 4-th power. This however comes at the cost of a sparsity requirement on the Hamiltonian and the input state, which can be restrictive in practical cases, i.e., in non-sparse but low-rank cases. Therefore, the results by [21] might generally better suitable for dense Hamiltonians. A big difference is that the algorithm of Tang and others are solving the same problem as the quantum algorithm that we developed, i.e., it allows us to sample from the output distribution of the time evolution. Taking this into account, we believe our classical Nyström based algorithm can be translated into this framework as well by using rejection sampling approaches similar to the ones introduced in [17]. However, we leave this as an open question for future work.

Summary of related results. To summarise this subsection, we provide Table 4.3 for state-of-the-art algorithms for Hamiltonian simulation which include quantum and classical ones.

4.4.3 Applications

In the following, we also lay out a few applications of Hamiltonian simulation. Since the main focus of this thesis is quantum machine learning we in particular refer to applications in this area. For this, we will discuss the quantum linear systems algorithm, which is a key subroutine in most of the existing quantum machine learning algorithms with an acclaimed exponential speedup. Of course, other applications such as the estimation of properties such as ground state energies of chemical systems exist [120].

Unitary implementation. One application which follows directly from the ability to simulate non-sparse Hamiltonians is the ability to approximately implement an arbitrary unitary matrix, the so-called unitary implementation problem: given access to the entries of a unitary U construct a quantum circuit $\tilde{U}$ such that $\left\| U - \tilde{U} \right\| \leq \varepsilon$ for some fixed ε. Unitary implementation can be reduced to Hamiltonian simulation as shown by Berry and others [81, 121]. For this, consider the

Model	State-of-the-art	Advantage of our quantum algorithm
Sparse-access with $\|H\|_{\max}$ dependence	$\widetilde{O}(ts\|H\|_{\max})$ [90]	No advantage
Sparse-access with $\|H\|$ dependence	$O((t\sqrt{s}\|H\|)^{1+o(1)}/\varepsilon^{o(1)})$ [92]	Subpolynomial improvement in t,s; exponential improvement in ε
qRAM	$\widetilde{O}(t\sqrt{s}\|H\|)$ [37]	Same result
Classical sampling and query access	$\mathrm{poly}(t,\|H\|_F,1/\varepsilon)$ [21]	Polynomial speedup
Classical sampling and query access, sparse input	$\mathrm{poly}(s,t,\|H\|_F,1/\varepsilon)$, [Thm.8]	Polynomial speedup

Figure 4.3: Comparing our result $O(t\sqrt{d}\,\|H\|\,\mathrm{polylog}(t,d,\|H\|,1/\varepsilon))$ with other quantum and classical algorithms for different models. Since the qRAM model is stronger than the sparse-access model and the classical sampling and query access model, we consider the advantage of our algorithm against others when they are directly applied to the qRAM model.

Hamiltonian of the form

$$H = \begin{pmatrix} 0 & U \\ U^H & 0 \end{pmatrix}. \tag{4.15}$$

By performing the time evolution according to $e^{-iH\pi/2}$ on the state $|1\rangle|\psi\rangle$, we are able to implement the operation $e^{-iH\pi/2}|1\rangle|\psi\rangle = -i|0\rangle U|\psi\rangle$, which means that we can apply U to $|\psi\rangle$. If the entries of U are stored in a data structure similar to Definition 2, using our Hamiltonian simulating algorithm we can implement the unitary U with time complexity (circuit depth) $O(\sqrt{N}\,\mathrm{polylog}(N,1/\varepsilon))$.

Quantum linear systems solver. As previously mentioned, one of the major subroutines of the quantum linear systems algorithm is Hamiltonian simulation. Hamiltonian simulation, in particular in combination with the phase estimation [122] algorithm is used to retrieve the eigenvalues of the input matrix, which can then be inverted to complete the matrix inverse. Assuming here for simplicity that $|b\rangle$ is

entirely in the column-space of A, then the linear systems algorithm solves a linear system of the form $Ax = b$ and outputs an approximation to the normalised solution $|x\rangle = |A^{-1}b\rangle$. For s-sparse matrices [93] showed that A^{-1} can be approximated as a linear combination of unitaries of the form e^{-iAt}. Notably for non-Hermitian A, we can just apply the same ideas to the Hamiltonian

$$H = \begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix}, \tag{4.16}$$

which is Hermitian by definition, and due to the logarithmic runtimes results in only a factor 2 overhead. In order to implement these unitaries we can then use any Hamiltonian simulation algorithm, such as [90], which results in an overall (gate) complexity of $O(s\kappa^2\text{polylog}(N, \kappa/\varepsilon))$. For a non-sparse input matrix A, the algorithm however scales as $\widetilde{O}(N)$ (again ignoring logarithmic factors). In some of our earlier results, we used a similar data structure to Definition 2 to design a quantum algorithm for solving linear systems for non-sparse matrices [94] which resulted in a time complexity (circuit depth) of $O(\kappa^2\sqrt{N}\text{polylog}(N)/\varepsilon)$. Using again this data structure, similarly to [16, 94], our new Hamiltonian simulation algorithm, and the linear combinations of unitaries (LCU) decompositions from [93], we can describe a quantum algorithm for solving linear systems for non-sparse matrices with time complexity (circuit depth)

$$O(\kappa^2\sqrt{N}\text{polylog}(\kappa/\varepsilon)),$$

which is an exponential improvement in the error dependence compared to our previous result [94]. Notably, one drawback of our implementation, which has been resolved by [37] and subsequent works is the high condition number dependency of our algorithm which seems restrictive for practical applications.

4.4.4 Hamiltonian Simulation for dense matrices

We now show how we can use the data structure from Definition 2 to derive a fast quantum algorithm for non-sparse Hamiltonian simulation. We will in particular

proof the results from Theorem 6.

We prove the result in multiple steps. First, we show how to use the data structure to prepare a certain state. Next, we use the state preparation to perform a quantum walk. In the final step, we show how the quantum walk can be used to implement Hamiltonian simulation. This then leads to the main result.

State Preparation. Using the data structure from Definition 2, we can efficiently perform the mapping described in the following technical lemma for efficient state preparation.

Lemma 7 (State Preparation). *Let $H \in \mathbb{C}^{N \times N}$ be a Hermitian matrix (where $N = 2^n$ for a n qubit Hamiltonian) stored in the data structure as specified in Definition 2. Each entry H_{jk} is represented with b bits of precision. Then the following holds*

1. *Let $\|H\|_1 = \max_j \sum_{k=0}^{N-1} |H_{jk}|$ as before. A quantum computer that has access to the data structure can perform the following mapping for $j \in \{0, \ldots, N-1\}$,*

$$|j\rangle |0^{\log N}\rangle |0\rangle \mapsto \frac{1}{\sqrt{\|H\|_1}} |j\rangle \sum_{k=0}^{N-1} |k\rangle \left(\sqrt{H_{jk}^*} |0\rangle + \sqrt{\frac{\|H\|_1 - \sigma_j}{N}} |1\rangle \right),$$
(4.17)

   with time complexity (circuit depth) $O(n^2 b^{5/2} \log b)$, where $\sigma_j = \sum_k |H_{jk}|$, and the square-root satisfies $\sqrt{H_{jk}} \left(\sqrt{H_{jk}^*} \right)^* = H_{jk}$.

2. *The size of the data structure containing all N^2 complex entries is $O(N^2 \log^2(N))$.*

In order to perform this mapping, we will need the following Lemma. We will use it in order to efficiently implement the conditional rotations of the qubits with complex numbers.

Lemma 8. *Let $\theta, \phi_0, \phi_1 \in \mathbb{R}$ and let $\widetilde{\theta}, \widetilde{\phi}_0, \widetilde{\phi}_1$ be the b-bit finite precision representation of θ, ϕ_0, and ϕ_1, respectively. Then there exists a unitary U that performs the*

following mapping:

$$U : |\widetilde{\phi}_0\rangle |\widetilde{\phi}_1\rangle |\widetilde{\theta}\rangle |0\rangle \mapsto |\widetilde{\phi}_0\rangle |\widetilde{\phi}_1\rangle |\widetilde{\theta}\rangle \left(e^{i\widetilde{\phi}_0} \cos(\widetilde{\theta}) |0\rangle + e^{i\widetilde{\phi}_1} \sin(\widetilde{\theta}) |1\rangle \right). \quad (4.18)$$

Moreover, U can be implemented with $O(b)$ 1- and 2-qubit gates.

Proof. Define U as

$$U = \left(\sum_{\widetilde{\phi}_0 \in \{0,1\}^b} |\widetilde{\phi}_0\rangle\langle\widetilde{\phi}_0| \otimes e^{i|0\rangle\langle 0|\widetilde{\phi}_0} \right) \left(\sum_{\widetilde{\phi}_1 \in \{0,1\}^b} |\widetilde{\phi}_1\rangle\langle\widetilde{\phi}_1| \otimes e^{i|1\rangle\langle 1|\widetilde{\phi}_1} \right)$$
$$\left(\sum_{\widetilde{\theta} \in \{0,1\}^b} |\widetilde{\theta}\rangle\langle\widetilde{\theta}| \otimes e^{-iY\widetilde{\theta}} \right), \quad (4.19)$$

where $Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$ is the Pauli Y matrix.

To implement the operator $\sum_{\widetilde{\theta} \in \{0,1\}^b} |\widetilde{\theta}\rangle\langle\widetilde{\theta}| \otimes e^{-iY\widetilde{\theta}}$, we use one rotation controlled on each qubit of the first register, with the rotation angles halved for each successive bit. The other two factors of U can be implemented in a similar way. Therefore, U can be implemented with $O(b)$ 1- and 2-qubit gates. $\qquad\square$

Before proving Lemma 7, we first describe the construction and the size of the data structure. Readers may refer to [16] for more details.

- The data structure is built from N binary trees $D_i, i \in \{0, \ldots, N-1\}$ and we start with an empty tree.

- When a new entry (i, j, H_{ij}) arrives, we create or update the leaf node j in the tree D_i, where the adding of the entry takes $O(\log(N))$ time, since the depth of the tree for $H \in \mathbb{C}^{N \times N}$ is at most $\log(N)$. Since the path from the root to the leaf is of length at most $\log(N)$ (under the assumption that $N = 2^n$), we have furthermore to update at most $\log(N)$ nodes, which can be done in $O(\log(N))$ time if we store an ordered list of the levels in the tree.

- The total time for updating the tree with a new entry is given by $\log(N) \times \log(N) = \log^2(N)$.

- The memory requirements for k entries are given by $O(k\log^2(N))$ as for every entry (j, k, H_{jk}) at least $\log(N)$ nodes are added and each node requires at most $O(\log(N))$ bits.

Now we are ready to prove Lemma 7.

Proof of Lemma 7. With this data structure, we can perform the mapping specified in Eq. (4.17), with the following steps. For each j, we start from the root of D_j. Starting with the initial state $|j\rangle\,|0^{\log N}\rangle\,|0\rangle$, first apply the rotation (according to the value stored in the root node and calculating the normalisation in one query) on the last register to obtain the state

$$\frac{1}{\sqrt{\|H\|_1}}\,|0\rangle\,|0^{\log N}\rangle\left(\sqrt{\sum_{k=0}^{N-1}|H_{jk}^*|}\,|0\rangle + \sqrt{\|H\|_1 - \sigma_j}\,|1\rangle\right), \qquad (4.20)$$

which is normalised since $\sigma_j = \sum_k |H_{jk}|$, and we have by definition that $\sqrt{\sum_{k=0}^{N-1}|H_{jk}^*|}\cdot\left(\sqrt{\sum_{k=0}^{N-1}|H_{jk}^*|}\right)^* = \sum_k |H_{jk}| = \sigma_j$. Then a sequence of conditional rotations is applied on each qubit of the second register to obtain the state as in Eq. (4.17). At level ℓ of the binary tree D_j, a query to the data structure is made to load the data c (stored in the node) into a register in superposition, the rotation to perform is proportional to $\left(\sqrt{c}, \sqrt{(\|H\|_1 - \sigma_j)/2^\ell}\right)$ (assuming at the root, $\ell = 0$, and for the leaves, $\ell = \log N$). Then the rotation angles will be determined by calculating the square root and trigonometric functions on the output of the query: this can be implemented with $O(b^{5/2})$ 1- and 2-qubit gates using simple techniques based on Taylor series and long multiplication as in [90], where the error is smaller than that caused by truncating to b bits. Then the conditional rotation is applied by the circuit described in Lemma 8, and the cost for the conditional rotation is $O(b)$. There are $n = \log(N)$ levels, so the cost excluding the implementation of the oracle is $O(nb^{5/2})$. To obtain quantum access to the classical data structure, a quantum addressing scheme is required. One addressing scheme described in [6] can be used. Although the circuit size of this addressing scheme is $\widetilde{O}(N)$ for each D_j, its circuit depth is $O(n)$. Therefore, the time complexity (circuit depth) for preparing

the state in Eq. (4.17) is $O(n^2 b^{5/2} \log n)$.

We use the following rules to determine the sign of the square-root of a complex number: if H_{jk} is not a negative real number, we write $H_{jk} = re^{i\varphi}$ (for $r \geq 0$ and $-\pi \leq \varphi \leq \pi$) and take $\sqrt{H_{jk}^*} = \sqrt{r}e^{-i\varphi/2}$; when H_{jk} is a negative real number, we take $\sqrt{H_{jk}^*} = \text{sign}(j-k)i\sqrt{|H_{jk}|}$ to avoid the sign ambiguity. With this recipe, we have $\sqrt{H_{jk}}\left(\sqrt{H_{jk}^*}\right)^* = H_{jk}$. $\qquad\qquad\qquad\square$

In order to convey the working of the data structure better, we give in the following a example of the state preparation procedure based on the data structure in Fig. 4.1. For the sake of comprehensibility and simplicity, we only take an example with 4 leaves $\{c_1, c_2, c_3, c_4\}$, and hence $H = [c_1\ c_2\ c_3\ c_4]^t$. The initial state (omitting the first register) is $|00\rangle |0\rangle$. Let $\sigma = |c_0| + |c_1| + |c_2| + |c_3|$. Apply the first rotation, we obtain the state

$$\frac{1}{\sqrt{\|H\|_1}} |00\rangle \left(\sqrt{|c_0| + |c_1| + |c_2| + |c_3|} |0\rangle + \sqrt{\|H\|_1 - \sigma_j} |1\rangle \right) =$$
$$\frac{1}{\sqrt{\|H\|_1}} |00\rangle \left(\sqrt{\sigma} |0\rangle + \sqrt{\|H\|_1 - \sigma} |1\rangle \right) =$$
$$\frac{1}{\sqrt{\|H\|_1}} \left(\sqrt{\sigma} |00\rangle |0\rangle + \sqrt{\|H\|_1 - \sigma} |00\rangle |1\rangle \right). \qquad (4.21)$$

Then, apply a rotation on the first qubit of the first register conditioned on the last register, we obtain the state

$$\frac{1}{\sqrt{\|H\|_1}} \left(\left(\sqrt{|c_0| + |c_1|} |00\rangle + \sqrt{|c_2| + |c_3|} |10\rangle \right) |0\rangle \right.$$
$$\left. + \left(\sqrt{\frac{\|H\|_1 - \sigma_j}{2}} (|00\rangle + |10\rangle) \right) |1\rangle \right). \qquad (4.22)$$

Next, apply a rotation on the second qubit of the first register conditioned on the

first qubit of the first register and last register, we obtain the desired state:

$$
\frac{1}{\sqrt{\|H\|_1}} \left(\sqrt{c_0} \, |00\rangle \, |0\rangle + \sqrt{c_1} \, |01\rangle \, |0\rangle + \sqrt{c_2} \, |10\rangle \, |0\rangle + \sqrt{c_3} \, |11\rangle \, |0\rangle \right.
$$

$$
+ \sqrt{\frac{\|H\|_1 - \sigma_j}{4}} \, |00\rangle \, |1\rangle + \sqrt{\frac{\|H\|_1 - \sigma_j}{4}} \, |01\rangle \, |1\rangle
$$

$$
\left. + \sqrt{\frac{\|H\|_1 - \sigma_j}{4}} \, |10\rangle \, |1\rangle + \sqrt{\frac{\|H\|_1 - \sigma_j}{4}} \, |11\rangle \, |1\rangle \right)
$$

$$
= \frac{1}{\sqrt{\|H\|_1}} \left(|00\rangle \left(\sqrt{c_0} \, |0\rangle + \sqrt{\frac{\|H\|_1 - \sigma_j}{4}} \, |1\rangle \right) \right.
$$

$$
+ |01\rangle \left(\sqrt{c_1} \, |0\rangle + \sqrt{\frac{\|H\|_1 - \sigma_j}{4}} \, |1\rangle \right)
$$

$$
+ |10\rangle \left(\sqrt{c_2} \, |0\rangle + \sqrt{\frac{\|H\|_1 - \sigma_j}{4}} \, |1\rangle \right)
$$

$$
\left. + |11\rangle \left(\sqrt{c_3} \, |0\rangle + \sqrt{\frac{\|H\|_1 - \sigma_j}{4}} \, |1\rangle \right) \right). \tag{4.23}
$$

The Quantum Walk Operator. Based on the data structure specified in Definition 2 and the efficient state preparation in Lemma 7, we construct a quantum walk operator for H as follows. First define the isometry T as

$$
T = \sum_{j=0}^{N-1} \sum_{b \in \{0,1\}} (|j\rangle\langle j| \otimes |b\rangle\langle b|) \otimes |\varphi_{jb}\rangle, \tag{4.24}
$$

with $|\varphi_{j1}\rangle = |0\rangle \, |1\rangle$ and

$$
|\varphi_{j0}\rangle = \frac{1}{\sqrt{\|H\|_1}} \sum_{k=0}^{N-1} |k\rangle \left(\sqrt{H_{jk}^*} \, |0\rangle + \sqrt{\frac{\|H\|_1 - \sigma_j}{N}} \, |1\rangle \right), \tag{4.25}
$$

where $\sigma_j = \sum_{k=0}^{N-1} |H_{jk}|$. Let S be the swap operator that maps $|j_0\rangle \, |b_0\rangle \, |j_1\rangle \, |b_1\rangle$ to $|j_1\rangle \, |b_1\rangle \, |j_0\rangle \, |b_0\rangle$, for all $j_0, j_1 \in \{0, \ldots, N-1\}$ and $b_0, b_1 \in \{0, 1\}$. We observe that

$$
\langle j| \langle 0| T^H S T |k\rangle |0\rangle = \frac{\sqrt{H_{jk}} \left(\sqrt{H_{jk}^*} \right)^*}{\|H\|_1} = \frac{H_{jk}}{\|H\|_1}, \tag{4.26}
$$

where the second equality is ensured by the choice of the square-root as in the proof of Lemma 7. This implies that

$$(I \otimes \langle 0|) T^H ST (I \otimes |0\rangle) = \frac{H}{\|H\|_1}. \tag{4.27}$$

The quantum walk operator U is defined as

$$U = iS(2TT^H - I). \tag{4.28}$$

A more general characterization of the eigenvalues of quantum walks is presented in [123]. Here we give a specific proof on the relationship between the eigenvalues of U and H as follows.

Lemma 9. *Let the unitary operator U be defined as in Eq. (4.28), and let λ be an eigenvalue of H with eigenstate $|\lambda\rangle$. It holds that*

$$U |\mu_\pm\rangle = \mu_\pm |\mu_\pm\rangle, \tag{4.29}$$

where

$$|\mu_\pm\rangle = (T + i\mu_\pm ST) |\lambda\rangle |0\rangle, \text{ and} \tag{4.30}$$

$$\mu_\pm = \pm e^{\pm i \arcsin(\lambda/\|H\|_1)}. \tag{4.31}$$

Proof. By the fact that $T^H T = I$ and $(I \otimes \langle 0|) T^H ST (I \otimes |0\rangle) = H / \|H\|_1$, and $(I \otimes \langle 1|) T^H ST (I \otimes |0\rangle) = 0$, we have

$$U |\mu_\pm\rangle = \mu_\pm T |\lambda\rangle |0\rangle + i \left(1 + \frac{2\lambda i}{\|H\|_1} \mu_\pm \right) ST |\lambda\rangle |0\rangle. \tag{4.32}$$

In order for this state being an eigenstate, it must hold that

$$1 + \frac{2\lambda i}{\|H\|_1} \mu_\pm = \mu_\pm^2, \tag{4.33}$$

and the solution is

$$\mu_{\pm} = \frac{\lambda i}{\|H\|_1} \pm \sqrt{1 - \frac{\lambda^2}{\|H\|_1^2}} = \pm e^{\pm i \arcsin(\lambda/\|H\|_1)}. \tag{4.34}$$

$\square$

Linear combination of unitaries and Hamiltonian simulation. Next, we need to convert the quantum walk operator $U = iS(2TT^H - I)$ into an operator for Hamiltonian simulation. For this, we consider the generating functions for the Bessel functions, denoted by $J_m(\cdot)$. From [124, (9.1.41)], we know that it holds that

$$\sum_{m=-\infty}^{\infty} J_m(z)\mu_{\pm}^m = \exp\left(\frac{z}{2}\left(\mu_{\pm} - \frac{1}{\mu_{\pm}}\right)\right) = e^{iz\lambda/\|H\|_1}, \tag{4.35}$$

where the second equality follows from Eq. (4.31) and the fact that $\sin(x) = (e^{ix} - e^{-ix})/2i$. This leads to the following linear combination of unitaries:

$$V_{\infty} = \sum_{m=-\infty}^{\infty} \frac{J_m(z)}{\sum_{j=-\infty}^{\infty} J_j(z)} U^m = \sum_{m=-\infty}^{\infty} J_m(z)U^m = e^{izH/\|H\|_1}, \tag{4.36}$$

where the second equality follows from the fact that $\sum_{j=-\infty}^{\infty} J_j(z) = 1$.

Since we cannot in practice implement the infinite sum, we will in the following find an approximation to $e^{-izH/\|H\|_1}$ by truncation the sum in Eq. (4.36):

$$V_k = \sum_{m=-k}^{k} \frac{J_m(z)}{\sum_{j=-k}^{k} J_j(z)} U^m. \tag{4.37}$$

Here the coefficients are normalised by $\sum_{j=-k}^{k} J_j(z)$ so that they sum to 1. This will minimize the approximation error (see the proof of Lemma 10), and the normalisation trick was originated in [90]). The eigenvalues of V_k are

$$\sum_{m=-k}^{k} \frac{J_m(z)}{\sum_{j=-k}^{k} J_j(z)} \mu_{\pm}^m. \tag{4.38}$$

Note that each eigenvalue of V_k does not depend on $\pm$ as $J_{-m}(z) = (-1)^m J_m(z)$.

To bound the error in this approximation, we require the following technical

lemma.

Lemma 10. *Let V_k and V_∞ be defined as above. There exists a positive integer k satisfying $k \geq |z|$ and*

$$k = O\left(\frac{\log(\|H\|/(\|H\|_1 \varepsilon))}{\log\log(\|H\|/(\|H\|_1 \varepsilon))}\right), \tag{4.39}$$

such that

$$\|V_k - V_\infty\| \leq \varepsilon. \tag{4.40}$$

Proof. The proof outlined here follows closely the proof of Lemma 8 in [90]. Recalling the definition of V_k and V_∞, we define the weights in V_k by

$$\alpha_m := \frac{J_m(z)}{C_k}, \tag{4.41}$$

where $C_k = \sum_{l=-k}^{k} J_l(z)$. The normalisation here is chosen so that $\sum_m a_m = 1$ which will give the best result [90].

Since

$$\sum_{m=-\infty}^{\infty} J_m(z) = \sum_{m=[-\infty:-k-1;k+1:\infty]} J_m(z) + \sum_{m=-k}^{k} J_m(z) = 1, \tag{4.42}$$

observe that we have two error sources. The first one comes from the truncation of the series, and the second one comes from the different renormalisation of the terms which introduces an error in the first $|m| \leq k$ terms in the sum. We therefore start by bounding the normalisation factor C_k. For the Bessel-functions for all m it holds that $|J_m(z)| \leq \frac{1}{|m|!}\left|\frac{z}{2}\right|^{|m|}$, since $J_{-m}(z) = (-1)^m J_m(z)$ [124, (9.1.5)]. For $|m| \leq k$ we can hence find the following bound on the truncated part

$$\sum_{m=[-\infty:-k-1;k+1:\infty]} |J_m(z)| = 2\sum_{m=k+1}^{\infty} |J_m(z)| \leq 2\sum_{m=k+1}^{\infty} \frac{|z/2|^m}{m!}$$

$$= 2\frac{|z/2|^{k+1}}{(k+1)!}\left(1 + \frac{|z/2|}{k+2} + \frac{|z/2|^2}{(k+2)(k+3)} + \cdots\right)$$

$$< 2\frac{|z/2|^{k+1}}{(k+1)!}\sum_{m=k+1}^{\infty}\left(\frac{1}{2}\right)^{m-(k+1)} = \frac{4|z/2|^{k+1}}{(k+1)!}. \tag{4.43}$$

Since $\sum_m J_m(z) = 1$, based on the normalisation, we hence find that

$$\sum_{m=-k}^{k} J_m(z) \geq \left(1 - \frac{4|z/2|^{k+1}}{(k+1)!}\right),\tag{4.44}$$

which is a lower bound on the normalisation factor C_k. Since $a_m = \frac{J_m(z)}{C_k}$, the correction is small, which implies that

$$a_m = J_m(z)\left(1 + O\left(\frac{|z/2|^{k+1}}{(k+1)!}\right)\right),\tag{4.45}$$

and we have a multiplicative error based on the renormalisation.

Next we want to bound the error in the truncation before we join the two error sources.

From Eq. (4.35) we know that

$$e^{iz\lambda/\Lambda} - 1 = \sum_{m=-\infty}^{\infty} J_m(z)(\mu_{\pm}^m - 1),\tag{4.46}$$

by the normalisation of $\sum_m J_m(z)$. From this we can see that we can hence obtain a bound on the truncated $J_m(z)$ as follows.

$$\sum_{m=-k}^{k} J_m(z)(\mu_{\pm}^m - 1) = e^{iz\lambda/\Lambda} - 1 - \sum_{\substack{m=[-\infty:-(k+1);\\(k+1):\infty]}} J_m(z)(\mu_{\pm}^m - 1).\tag{4.47}$$

Therefore we can upper bound the left-hand side in terms of the exact value of V_∞, i.e. $e^{iz\lambda/\Lambda}$ if we can bound the right-most term in Eq. (4.47). Using furthermore the bound in Eq. (4.45) we obtain

$$\sum_{m=-k}^{k} a_m(z)(\mu_{\pm}^m - 1) = \left(e^{iz\lambda/\Lambda} - 1 - \sum_{\substack{m=[-\infty:-(k+1);\\(k+1):\infty]}} J_m(z)(\mu_{\pm}^m - 1)\right)$$
$$\left(1 + O\left(\frac{|z/2|^{k+1}}{(k+1)!}\right)\right),\tag{4.48}$$

which reduced with $\left| 2^{iz\lambda/\Lambda} - 1 \right| \leq |z\lambda/\Lambda|$ and $|z| \leq k$ to

$$\sum_{m=-k}^{k} a_m(z)(\mu_{\pm}^m - 1) = e^{iz\lambda/\Lambda} - 1 - O\left(\sum_{\substack{m=[-\infty:-(k+1); \\ (k+1):\infty]}} J_m(z)(\mu_{\pm}^m - 1) \right). \quad (4.49)$$

We can then obtain the desired bound $\|V_\infty - V_k\|$ by reordering the above equation, and using that $\sum_{m=-k}^{k} a_m(z) = 1$ such that we have

$$\|V_\infty - V_k\| = \left| \sum_{m=-k}^{k} a_m \mu_{\pm}^m - e^{iz\lambda/\Lambda} \right| = O\left(\sum_{\substack{m=[-\infty:-(k+1); \\ (k+1):\infty]}} J_m(z)(\mu_{\pm}^m - 1) \right). \quad (4.50)$$

We hence only need to bound the right-hand side.

For μ_+ we can use that $|\mu_+^m - 1| \leq 2|m\lambda/\Lambda| =: 2|mv|$ and obtain the bound $2\frac{|v|}{k!}\left|\frac{z}{2}\right|^{k+1}$ [90]. For the μ_- case we need to refine the analysis and will show that the bound remains the same. Let $v := \lambda/\Lambda$ as above. First observe that $J_m(z)\mu_-^m + J_{-m}(z)\mu_i^{-m} = J_{-m}(z)\mu_+^{-m} + J_m(z)\mu_+^m$, and it follows that

$$\sum_{m=-\infty}^{-(k+1)} J_m(z)(\mu_-^m - 1) + \sum_{m=k+1}^{\infty} J_m(z)(\mu_-^m - 1)$$
$$= \sum_{m=-\infty}^{-(k+1)} J_m(z)(\mu_+^m - 1) + \sum_{m=k+1}^{\infty} J_m(z)(\mu_+^m - 1). \quad (4.51)$$

Therefore we only need to treat the μ_+ case.

$$
\left| \sum_{\substack{m=[-\infty:-(k+1); \\ (k+1):\infty]}} J_m(z)(\mu_+^m - 1) \right| \leq 2 \left| \sum_{m=k+1}^{\infty} J_m(z)(\mu_+^m - 1) \right|
$$

$$
\leq 2 \sum_{m=k+1}^{\infty} |J_m(z)||\mu_+^m - 1|
$$

$$
= 2 \sum_{m=k+1}^{\infty} \frac{1}{|m|!} \left| \frac{z}{2} \right|^{|m|} |\mu_+^m - 1|
$$

$$
\leq 4 \sum_{m=k+1}^{\infty} \frac{1}{|m|!} \left| \frac{z}{2} \right|^{|m|} m|\nu|
$$

$$
< \frac{8|\nu|}{(k+1)!} \left| \frac{z}{2} \right|^{k+1} (k+2). \tag{4.52}
$$

Using this bound, we hence obtain from Eq. (4.50),

$$
\|V_\infty - V_k\| = \left| \sum_{m=-k}^{k} a_m \mu_\pm^m - e^{iz\lambda/\Lambda} \right| \leq O\left(\frac{\lambda}{k!\,\Lambda} \left| \frac{z}{2} \right|^{k+1} \right) = O\left(\frac{\|H\|\,(z/2)^{k+1}}{\Lambda k!} \right). \tag{4.53}
$$

In order for the above equation being upper-bounded by ε, it suffices to choose some k that is upper bounded as claimed. $\qquad \square$

As we can see from the above discussion, we can hence use the operator V_k to implement the time evolution according to $e^{-izH/\|H\|_1}$. We can also immediately see from this, that V_k is a linear combination of unitaries (LCU). We proceed now to implement this LCU by using some well-known results.

In the following, we provide technical lemmas for implementing linear combination of unitaries. Suppose we are given the implementations of unitaries U_0, U_1, ..., U_{m-1}, and coefficients $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$. Then the unitary

$$
V = \sum_{j=0}^{m-1} \alpha_j U_j \tag{4.54}
$$

can be implemented probabilistically by the technique called linear combination of unitaries (LCU) [125]. Provided $\sum_{j=0}^{m-1} |\alpha_j| \leq 2$, V can be implemented with success probability $1/4$. To achieve this, we define the multiplexed-U operation, which is

denoted by multi-U, as

$$\text{multi-}U \, |j\rangle \, |\psi\rangle = |j\rangle \, U_j \, |\psi\rangle . \tag{4.55}$$

The probabilistic implementation of V is summarised in the following lemma.

Lemma 11. *Let* multi-U *be defined as above. If* $\sum_{j=0}^{m-1} |\alpha_j| \leq 2$, *then there exists a quantum circuit that maps* $|0\rangle \, |0\rangle \, |\psi\rangle$ *to the state*

$$\frac{1}{2} \, |0\rangle \, |0\rangle \left(\sum_{j=0}^{m-1} \alpha_j U_j \, |\psi\rangle \right) + \frac{\sqrt{3}}{2} |\Phi^\perp\rangle , \tag{4.56}$$

where $(|0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes I) \, |\Phi^\perp\rangle = 0$. *Moreover, this quantum circuit uses* $O(1)$ *applications of* multi-U *and* $O(m)$ *1- and 2-qubit gates.*

Proof. Let $s = \sum_{j=0}^{m-1} |\alpha_j|$. We first define the unitary operator B to prepare the coefficients:

$$B \, |0\rangle \, |0\rangle = \left(\sqrt{\frac{s}{2}} \, |0\rangle + \sqrt{1 - \frac{s}{2}} \, |1\rangle \right) \otimes \frac{1}{\sqrt{s}} \sum_{j=0}^{m-1} \sqrt{\alpha_j} \, |j\rangle . \tag{4.57}$$

Define the unitary operator W as $W = (B^H \otimes I)(I \otimes \text{multi-}U)(B \otimes I)$. We claim that W performs the desired mapping, as

$$
\begin{aligned}
W \, |0\rangle \, |0\rangle \, |\psi\rangle = & (B^H \otimes I)(I \otimes \text{multi-}U)(B \otimes I) \, |0\rangle \, |0\rangle \, |\psi\rangle \\
= & \frac{1}{\sqrt{2}} (B^H \otimes I) \, |0\rangle \sum_{j=0}^{m-1} \sqrt{\alpha_j} \, |j\rangle \, U_j \, |\psi\rangle \\
& + \sqrt{\frac{2-s}{2s}} (B^H \otimes I) \, |1\rangle \sum_{j=0}^{m-1} \sqrt{\alpha_j} \, |j\rangle \, U_j \, |\psi\rangle \\
= & \frac{1}{2} \, |0\rangle \, |0\rangle \sum_{j=0}^{m-1} \alpha_j U_j \, |\psi\rangle + \sqrt{\gamma} |\Phi^\perp\rangle , \tag{4.58}
\end{aligned}
$$

where $|\Phi^\perp\rangle$ is a state satisfying $(|0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes I) \, |\Phi^\perp\rangle = 0$, and γ is some normalisation factor.

The number of applications of multi-U is constant, as in the definition of W.

To implement the unitary operator B, $O(m)$ 1- and 2-qubit gates suffice. $\qquad\square$

Let W be the quantum circuit in Lemma 11, and let P be the projector defined as $P = |0\rangle\langle0| \otimes |0\rangle\langle0| \otimes I$. We have

$$PW\,|0\rangle\,|0\rangle\,|\psi\rangle = \frac{1}{2}\,|0\rangle\,|0\rangle\sum_{j=0}^{m-1}\alpha_j U_j\,|\psi\rangle. \qquad (4.59)$$

If $\sum_{j=0}^{m-1}\alpha_j U_j$ is a unitary operator, one application of the oblivious amplitude amplification operator $-W(1-2P)W^H(1-2P)W$ implements $\sum_{j=0}^{m-1}U_j$ with certainty [82]. However, in our application, the unitary operator $\widetilde{W}$ implements an approximation of V_∞ in the sense that

$$P\widetilde{W}\,|0\rangle\,|0\rangle\,|\psi\rangle = \frac{1}{2}\,|0\rangle\,|0\rangle\,V_k\,|\psi\rangle, \qquad (4.60)$$

with $\|V_k - V_\infty\| \leq \varepsilon$. The following lemma shows that the error caused by the oblivious amplitude amplification is bounded by $O(\varepsilon)$.

Lemma 12. *Let the projector P be defined as above. If a unitary operator $\widetilde{W}$ satisfies $P\widetilde{W}\,|0\rangle\,|0\rangle\,|\psi\rangle = \frac{1}{2}\,|0\rangle\,|0\rangle\,\widetilde{V}\,|\psi\rangle$ where $\left\|\widetilde{V} - V\right\| \leq \varepsilon$. Then $\left\|-\widetilde{W}(I-2P)\widetilde{W}^H(I-2P)\widetilde{W}\,|0\rangle\,|0\rangle\,|\psi\rangle - |0\rangle\,|0\rangle\,V\,|\psi\rangle\right\| = O(\varepsilon)$.*

Proof. We have

$$-\widetilde{W}(I-2P)\widetilde{W}^H(I-2P)\widetilde{W}\,|0\rangle\,|0\rangle\,|\psi\rangle$$
$$= (\widetilde{W} + 2P\widetilde{W} - 4\widetilde{W}P\widetilde{W}^H P\widetilde{W})\,|0\rangle\,|0\rangle\,|\psi\rangle$$
$$= (\widetilde{W} + 2P\widetilde{W} - 4\widetilde{W}P\widetilde{W}^H PP\widetilde{W}P)\,|0\rangle\,|0\rangle\,|\psi\rangle$$
$$= \widetilde{W}\,|0\rangle\,|0\rangle\,|\psi\rangle + |0\rangle\,|0\rangle\,\widetilde{V}\,|\psi\rangle - \widetilde{W}\left(|0\rangle\,|0\rangle\,\widetilde{V}^H\widetilde{V}\,|\psi\rangle\right) \qquad (4.61)$$

Because $\left\|\widetilde{V} - V\right\| \leq \varepsilon$ and V is a unitary operator, we have $\left\|\widetilde{V}^H\widetilde{V} - I\right\| = O(\varepsilon)$. Therefore, we have

$$\left\|-\widetilde{W}(I-2P)\widetilde{W}^H(I-2P)\widetilde{W}\,|0\rangle\,|0\rangle\,|\psi\rangle - |0\rangle\,|0\rangle\,\widetilde{V}\,|\psi\rangle\right\| = O(\varepsilon). \qquad (4.62)$$

Thus

$$\left\| -\widetilde{W}(I-2P)\widetilde{W}^H(I-2P)\widetilde{W} \,|0\rangle\,|0\rangle\,|\psi\rangle - |0\rangle\,|0\rangle\,V\,|\psi\rangle \right\| = O(\varepsilon). \qquad (4.63)$$

$\square$

Now we are ready to prove Theorem 6.

Proof of Theorem 6. The proof we outline here follows closely the proof given in [90]. The intuition of this algorithm is to divide the simulation into $O(t\,\|H\|_1)$ segments, with each segment simulating $e^{-iH/2}$. To implement each segment, we use the LCU technique to implement V_k defined in Eq. (4.37), with coefficients $\alpha_m = J_m(z)/\sum_{j=-k}^{k} J_j(z)$. When $z = -1/2$, we have $\sum_{j=-k}^{k}|\alpha_j| < 2$. Actually, this holds for all $|z| \le 1/2$ because

$$\sum_{j=-k}^{k}|\alpha_j| \le \sum_{j=-k}^{k} \frac{|J_j(z)|}{1-4\frac{|z/2|^{k+1}}{(k+1)!}} \le \sum_{j=-k}^{k} \frac{|z/2|^{|j|}}{|j|!}\left(1 - \frac{4|z/2|^{k+1}}{(k+1)!}\right)^{-1}$$

$$< \frac{8}{7} + \frac{16}{7}\sum_{j=1}^{\infty}\frac{1}{4^j j!} = \frac{8}{7} + \frac{16}{7}\left(\sqrt[4]{e}-1\right) < 2, \qquad (4.64)$$

where the first inequality follows from the fact that $\sum_{j=-k}^{k} J_j(z) \ge 1-4|z/2|^{k+1}/(k+1)!$ (see [90]), the second inequality follows from the fact that $|J_m(z)| \le |z/2|^{|m|}/|m|!$ (see [124, (9.1.5)]), and the third inequality uses the assumption that $|z| \le 1/2$. Now, Lemmas 11 and 12 can be applied. By Eq. (4.35) and using Lemma 10, set

$$k = O\left(\frac{\log(\|H\|/(\|H\|_1\varepsilon'))}{\log\log(\|H\|/(\|H\|_1\varepsilon'))}\right), \qquad (4.65)$$

and we obtain a segment that simulates $e^{-iH/(2\|H\|_1)}$ with error bounded by $O(\varepsilon')$. Repeat the segment $O(t\,\|H\|_1)$ times with error $\varepsilon' = \varepsilon/(t\,\|H\|_1)$, and we obtain a simulation of e^{-iHt} with error bounded by ε. It suffices to take

$$k = O\left(\frac{\log(t\,\|H\|/\varepsilon)}{\log\log(t\,\|H\|/\varepsilon)}\right). \qquad (4.66)$$

By Lemma 11, each segment can be implemented by $O(1)$ application of

multi-U and $O(k)$ 1- and 2-qubit gates, as well as the cost for computing the co-efficients α_m for $m \in \{-k, \ldots, k\}$. The cost for each multi-U is k times the cost for implementing the quantum walk U. By Lemma 7, the state in Eq. (4.25) can be prepared with time complexity (circuit depth) $O(n^2 b^{5/2})$, where b is the number of bit of precision. To achieve the overall error bound ε, we choose $b = O(\log(t \|H\|_1 / \varepsilon))$. Hence the time complexity for the state preparation is $O(n^2 \log^{5/2}(t \|H\|_1 / \varepsilon))$, which is also the time complexity for applying the quantum walk U. Therefore, the time complexity for one segment is

$$O\left(n^2 \log^{5/2}(t \|H\|_1 / \varepsilon) \frac{\log(t \|H\| / \varepsilon)}{\log \log(t \|H\| / \varepsilon)}\right). \tag{4.67}$$

Considering $O(t \|H\|_1)$ segments, the time complexity is as claimed. $\qquad\square$

Note that the coefficients $\alpha_{-k}, \ldots, \alpha_k$ (for k defined in Eq. (4.65)) in Lemma 11 can be classically computed using the methods in [126, 127], and the cost is $O(k)$ times the number of bits of precision, which is $O(\log(t \|H\| / \varepsilon)$. This is no larger than the quantum time complexity.

Discussion. We have hence seen that we can design a quantum Hamiltonian simulation algorithm which hast time complexity $\widetilde{O}(\sqrt{N})$ even for non-sparse Hamiltonians. The algorithm relies heavily on the access to a seemingly powerful input model. While it is questionable that it is even possible to implement such a data structure physically due to the exponential amount of quantum resources [43, 72, 5], and this requirement might even be further increased through a potentially strong require-ments in terms of hte error rate per gate of $O(1/\text{poly}(N))$ to retain a feasible error rate for applications [75], for us there is an even more important question: How fast can classical algorithms be if they are given a similarly powerful data structure? We investigate this question in the next section, where we use the so-called Nyström approach to simulate a Hamiltonian on a classical computer.

4.4.5 Hamiltonian Simulation with the Nyström method

In this section, we derive a classical, randomised algorithm for the strong simulation of quantum Hamiltonian dynamics which is based on the Nyström method that we

introduced in Section 4.4.2[Quantum-inspired classical algorithms for Hamiltonian simulation]. We particularly prove the results in Theorem 8, and the Corollary 9. For this, we develop an algorithm for so-called strong quantum simulation, where our objective is to obtain an algorithm which can compute the amplitude of a particular outcome and hence the entries of the final state after the evolution for time t. This is in contrast to the weak simulation, where we require the algorithm to only be able to sample from the output distribution of a quantum circuit. Informally, in the one case we are hence able to query the algorithm with an index i, and the algorithm will be able to return $\psi[i]$ (or more precisely the projection of the state vector in the i-th element of the computational basis) while in the other case we can only obtain the basis state i with probability $|\psi[i]|^2$. Strong simulation is known to have unconditional and exponential lower bounds [128]. This implies that it is therefore in general hard for both classical and quantum computers. On the other hand, weak simulation can be performed efficiently by a quantum computer (for circuits of polynomial size). From the perspective of computational complexity, we are therefore attempting to solve a stronger problem compared to the Hamiltonian simulation that the quantum algorithm performs, since the latter can only sample from the probability distribution induced by measurements on the output state. Surprisingly, we are still able to find cases in which the time evolution can be simulated efficiently. Our algorithm is able to efficiently perform the Hamiltonian simulation and output the the requested amplitude (c.f., Theorem 8) if we grant it access to a memory structure which fulfills the following requirements. Note that these are in principle more general than the assumption of the classical memory structure from Definition 3 but the memory structure immediately fulfills these requirements. On the other hand, we also require input and row sparsity in order to perform the simulation efficiently (i.e., in $O(\log(N))$ time for a n-qubit system with dimension $N = 2^n$. We next discuss these requirements in more detail.

Input Requirements We assume throughout the chapter that the input matrix and state are sparse, i.e., we assume that every row of H has at most s non-zero entries, and that the input state ψ has at most q non-zero entries. In order for algorithm to

work, we then require the following assumptions:

1. We require H to be *row-computable*, i.e., there exists a classical efficient algorithm that, given a row-index i, outputs a list of the non-zero entries (and their indices) of the row. If we are not having access to a memory structure such as described in Def. 3 or a structured Hamiltonian, then this condition is in general only fulfilled if every row of H has at most a number $s = O(\text{polylog}(N))$ of non-zero entries.

2. We require that the entries of the initial state ψ are row-computable, i.e., there exists a classical efficient algorithm that outputs a list of the non-zero entries similar as above. In order for this to be generally valid, we require the state to have at most $q = O(\text{polylog}(N))$ non-zero entries. However, we can also use the data structure given in Def. 3 or a structured input.

3. We require H to be efficiently *row-searchable*. This condition informally states that we can efficiently sample randomly selected indices of the rows of H in a way proportional to the norm of the row (general case) or the diagonal element of the row (positive semidefinite case). This is fundamentally equivalent to the sample-and-query access discussed earlier, and indeed, the data structure allows us to immediately perform this operation as well. We will show this relationship later on.

While the notion of row-searchability is commonly assumed to hold in the randomised numerical linear algebra literature (e.g., [129][Section 4]), in the context of quantum systems this is not given in general, since we are dealing with exponentially sized matrices. It is of course reasonable to assume that for a polynomially sized matrix we are indeed able to evaluate all the row-norms efficiently in a time (number of steps) proportional to the number of non-zero entries.

In order to obtain an efficient algorithm (i.e., one which only depends logarithmically on the dimension $N = 2^n$ for a n-qubit system), we require sparsity of the Hamiltonian H and the input state ψ. In general we therefore require

$s = q = O(\text{polylog}(N))$ for the non-zero entries in order for the algorithm to be efficient.

High-level description of the algorithm The algorithm proceeds by performing a two steps approximation. First, we approximate the Hamiltonian H in terms of a low rank operator $\widehat{H}$ which is small, and therefore more amendable for the computations which we perform next. We obtain $\widehat{H}$ by sampling the rows with a probability proportional to the row norm, and hence do rely on a Nyström scheme. Second, we approximate the time evolution of the input state $e^{i\widehat{H}t}\psi$ via a truncated Taylor expansion of the matrix exponential. This can be efficiently performed since we can use the small operator $\widehat{H}$ and the spectral properties of the truncated exponential..

As an addition, we separately consider the case of a generic Hamiltonian H and the restricted case of positive semidefinite Hamiltonians. As is the case in general, for the more restricted case of the PSD Hamiltonian, we are able to derive tighter bounds.

We will briefly discuss the sampling scheme to obtain $\tilde{H}$ to give the reader an idea of the overall procedure before going into the detailed proofs.

In both cases, our algorithm leverages on a low-rank approximation of the Hamiltonian H to efficiently approximate the matrix exponential e^{iHt} which are obtained by randomly sampling $M = O(\text{polylog}(N))$ rows according to the magnitude of the row norms, and then collating them in a matrix $A \in \mathbb{C}^{M \times N}$. Let now $A \in \mathbb{C}^{M \times N}$ be the matrix obtained by sub-sampling M rows of H, and $B \in \mathbb{C}^{M \times M}$ the matrix obtained by selecting the columns of A whose indices correspond to those M indices for the rows of H.

For SPD H, we then use an approximation of the form $\widehat{H} = AB^+A^H$, where B^+ is the pseudoinverse of the SPD matrix B.

Next, in order to perform the time evolution $e^{i\widehat{H}t}\psi$ by truncating the Taylor series expansion of the matrix exponential function after the K-th order. In order to apply the individual terms in the truncated Taylor series, we then make use of the structure of $\widehat{H}$, and formulate the operator only in terms of linear operations involving the matrices A^HA, B^+ and B and the vector $A^H\psi$. Under the above assumptions

(Paragraph 4.4.5[Input Requirements]), and for $s = q = O(\text{polylog}(N))$ all these operations can be performed efficiently.

In the general Hermitian setting, we form A by first sampling the rows of H and then rescale the sampled rows according to their sampling probability. Unlike in the case of SPD H, we approximate the Hamiltonian by the approximation $\widehat{H}^2 := AA^H$ to approximate H^2. This approximation is useful here, since we decompose the matrix exponential into two auxiliary functions, and then for each of these evaluate its truncated Taylor series expansion. By doing so, we can formulate the final approximation solely in terms of linear operations involving $A^H A$ and $A^H \psi$. These operations can then again be performed efficiently under the initial assumptions oh H and ψ.

Row-searchability implies efficient row-sampling As we see in the assumptions, all our algorithms require the Hamiltonian to be row-searchable. In this section we describe an efficient algorithm for sampling rows of a row-searchable Hamiltonians according to some probability distribution. Let $n \in \mathbb{N}$. We first introduce a binary tree of subsets spanning $\{0,1\}^n$. In the following, with abuse of notation, we identify binary tuples with the associated binary number. Let L be a binary string with $|L| \leq n$, where $|L|$ denotes the length of the string. We denote with $S(L)$ the set

$$S(L) = \{L\} \times \{0,1\}^{n-|L|} = \{(L_1, \ldots, L_{|L|}, v_1, \ldots, v_{n-|L|}) \mid v_1, \ldots, v_{n-|L|} \in \{0,1\}\}. \tag{4.68}$$

We are now ready to state the row-searchability property for a matrix H.

Definition 4 (Row-searchability)**.** *Let H be a Hermitian matrix of dimension 2^n, for $n \in \mathbb{N}$. H is* row-searchable *if, for any binary string L with $|L| \leq n$, it is possible to compute the following quantity in $O(\text{poly}(n))$*

$$w(S(L)) = \sum_{i \in S(L)} h(i, H_{:,i}), \tag{4.69}$$

where h is the function computing the weight associated to the i-th column $H_{:,i}$. For positive semidefinite H we use $h(i, H_{:,i}) = H_{i,i}$, i.e. the diagonal element i while for

general Hermitian H we use $h(i, H_{:,i}) = \|H_{:,i}\|^2$.

Row-searchability intuitively works as follows: We are working with the following binary tree corresponding to a Hamiltonian H. We start at the leaves of the tree, which contain the individual probabilities according to which we want to sample from H. The parents at each level are then given (and computed) by the marginals over their children nodes, i.e., the sum over the probabilities of the children (assuming discrete probabilities). Using this tree, we can for a randomly sampled number in $[0, 1]$ traverse through the levels of the tree in $\log(N)$ time to find the leave node that is sampled, i.e. the indices of the column of H. More specifically, row-searchability then requires the evaluation of $w(S(L)))$ as defined in Eq. (4.69) which computes marginals of the diagonal of H or the norm $\|H_{:,i}\|^2$ in the general case, where the co-elements, *i.e.* the elements where we are not summing over, are defined by the tuple L. Hence, for empty L, $w(S(L)) = \text{Tr}(H)$. Note that this assumption is obviously closely related to the data structure given in Definition 3, and indeed we can use this data structure to immediately perform these operations.

Algorithm 3 MATLAB code for the sampling algorithm

Input: `wS(L)` corresponds to the function $w(S(L))$ defined in Eq. 4.69.
Output: L is the sampled row index

```
L = [];
q = rand()*wS(L);

for i=1:n
  if q >= wS([L 0])
      L = [L 0];
  else
      L = [L 1];
      q = q - wS([L 0]);
  end
end
```

Note that the function h that we described above is indeed related to leverage score sampling, which is a widely used sampling process in randomised numerical linear algebra [**?**, 69]. Leverage scores allow us to efficiently obtain sample prob-

abilities which have a sufficiently low variance to obtain fast algorithms with low error.

Alg. 3 describes an algorithm, that, given a row-searchable H, is able to sample an index with probability $p(j) = h(j, H_{:,j})/w(\{0,1\}^n)$. Let q be a random number uniformly sampled in $[0,T]$, where $T = w(\{0,1\}^n)$ is the sum of the weights associated to all the rows. The algorithm uses logarithmic search, starting with L empty and adding iteratively 1 or 0, to find the index L such that $w(\{0,\ldots,L-1\}) \leq q \leq w(\{0,\ldots,L\})$. The total time required to compute one index, is $O(nQ(n))$ where $Q(n)$ is the maximum time required to compute a $w(S(L))$ for $L \in \{0,1\}^n$. Note that if $w(S(L))$ can be computed efficiently for any $L \in \{0,1\}^n$, then $Q(n)$ is polynomial and the cost of the sampling procedure will be polynomial.

Remark 1 (Row-searchability more general than sparsity). *Note that if H has a polynomial number of non-zero elements, then $w(S(L))$ can be always computed in polynomial time. Indeed given L, we go through the list of elements describing H and select only the ones whose row-index starts with L and then compute $w(S(L))$, both step requiring polynomial time. However $w(S(L))$ can be computed efficiently even for Hamiltonians that are not polynomially sparse. For example, take the diagonal Hamiltonian defined by $H_{ii} = 1/i$ for $i \in [2^n]$. This H is not polynomially sparse and in particular it has an exponential number of non-zero elements, but still $w(S(L))$ can be computed in polynomial time, here in particular in $O(1)$. As it turns out, the data structure from Definition 3, which was first proposed by Tang [17] to do a similar sampling process efficiently, also allows us to perform this operation efficiently with the difference, that this holds then true for arbitrary Hamiltonians.*

Algorithm for PSD row-searchable Hermitian matrices Given a $2^n \times 2^n$ (i.e., $N \times N$) matrix $H \succeq 0$, the algorithm should output an approximation for the state given by the time evolution

$$\psi(t) = \exp(iHt)\psi. \tag{4.70}$$

The algorithm will do this through an expression of the form $\widehat{\exp}(i\widehat{H}t)\psi$, where $\widehat{\exp}$ and $\widehat{H}$ are an approximation of the exponential function and the low rank approximation of H respectively.

The first algorithm we describe here applies for for $H \succeq 0$. We will then generalise this result in the following section to arbitrary Hermitian H. All our results hold under the row-searchability condition, i.e. if condition 4 is fulfilled.

Let h be the diagonal of the positive semidefinite H and let $t_1, \ldots, t_M$, with $M \in \mathbb{N}$ be indices i.i.d. sampled with repetition from $\{1, \ldots, 2^n\}$ according to the probabilities

$$p(q) = h_q / \sum_i h_i, \tag{4.71}$$

e.g. via Alg. 3. Then, let $B \in \mathbb{C}^{M \times M}$ where $B_{i,j} = H_{t_i, t_j}$, for $1 \leq i, j \leq M$. Let furthermore $A \in \mathbb{C}^{2^n \times M}$ be the matrix with $A_{i,j} = H_{i,t_j}$ for $1 \leq i \leq 2^n$ and $1 \leq j \leq M$. We then define the approximation for H by $\widehat{H} = AB^+A^H$, where $(\cdot)^+$ denotes again the pseudoinverse. We now define a function $g(x) = (e^{itx} - 1)/x$. We can reformulate this, and immediately have $e^{itx} = 1 + g(x)x$, and note that g is an analytic function, i.e., it has a series expansion

$$g(x) = \sum_{k \geq 1} \frac{(it)^k}{k!} x^{k-1}.$$

Then

$$e^{i\widehat{H}t} = I + g(\widehat{H})\widehat{H} = I + g(AB^+A^H)AB^+A^H = I + Ag(B^+A^HA)B^+A^H, \tag{4.72}$$

where the last step is due to the fact that for any analytic function $q(x) = \sum_{k \geq 0} \alpha_k x^k$, it holds that

$$q(AB^+A^H)AB^+A^H = \sum_{k \geq 1} \alpha_k (AB^+A^H)^k AB^+A^H$$

$$= A \sum_{k \geq 1} \alpha_k (B^+A^HA)^k B^+A^H = Aq(B^+A^HA)B^+A^H.$$

By writing $D = B^+A^HA$, the algorithm then performs the operation

$$\widehat{\psi}_M(t) = \psi + Ag(D)B^+A^H\psi.$$

Next, in order to make the computation feasible, we truncate the series expansion after a finite number of terms. To do this, we hence approximate g with $g_K(x)$, which limits the series defining g to the first K terms, for $K \in \mathbb{N}$. Moreover, we can evaluate the function $g_K(D)B^+(A^H\psi)$ in an iterative fashion, and for this chose

$$b_j = \frac{(it)^{K-j}}{(K-j)!}v + Db_{j-1}, \quad v = B^+(A^H\psi),$$

where $b_0 = \frac{(it)^K}{K!}v$ and so $b_{K-1} = g(D)B^+A^H\psi$. Then, the new approximate state is given by

$$\widehat{\psi}_{K,M}(t) = \psi + Ab_{K-1}. \tag{4.73}$$

We summarise the algorithm in form of a MATLAB implementation in Alg. 4. Next, we analyse the cost of this algorithm. Let the row sparsity s of H be of order poly(n). Then the total cost of applying this operator is given by $O(M^2\text{poly}(n) + KM^2 + M^3))$ time complexity, where the terms M^3 and $M^2\text{poly}(n)$ are resulting from the calculation of D and the inverse. To compute the total cost in terms of space/memory, note that we do not have to save H or A in memory, but only B, D and the vectors v, b_j, which requires a total cost of $O(M^2)$. Indeed D can be computed in the following way: Assuming, without loss of generality, to have $2^n/M \in \mathbb{N}$, then

$$D = B^{-1} \sum_{i=1}^{2^n/M} A^H_{M(i-1)+1:Mi}A_{M(i-1)+1:Mi},$$

where $A_{a:b}$ is the submatrix of A containing the rows from a to b. A similar reasoning holds for the computation of the vector v. In the above computation we have assumed that we can efficiently sample from the matrix H according to the probabilities in Eq. 4.71. In order to do this, we are relying on the sampling algorithm which is summarised in Algorithm 3.

Algorithm 4 MATLAB code for approximating Hamiltonian dynamics when H is PSD

Input: $M, T = t_1, \ldots, t_M$ list of indices computed via Alg. 3. The function `compute_H_subMatrix`, given two lists of indices, computes the associated submatrix of H. `compute_psi_subVector`, given a list of indices, computes the associated subvector of ψ.
Output: vector b, s.t. $b = e^{iHt}\psi$.

```
B = compute_H_subMatrix(T, T);

D = zeros(M,M);
v = zeros(M,1);
for i=1:(2^n/M)
    E = compute_H_subMatrix((i-1)*M+1:M*i, T);
    D = D + E'*E;
    v = v + E'*compute_Psi_subVector((i-1)*M+1:M*i);
end
u
D = B\D;

b = zeros(M,1);
for j=1:K
    b = (1i*t)^(K-j)/factorial(K-j) * v + D*b;
end
```

We next analyse the errors and complexity of the algorithm in more detail, i.e., we derive bounds on K and M for a concrete approximation error ε. The results are summarised in the following theorem, which hold for the SPD case of H.

Theorem 10 (Algorithm for simulating PSD row-searchable Hermitian matrices)**.** *Let $\varepsilon, \delta \in (0,1]$, let $K, M \in \mathbb{N}$ and $t > 0$. Let H be positive semidefinite, where K is the number of terms in the truncated series expansions of $g(\hat{H})$ and M the number of samples we take for the approximation. Let $\psi(t)$ be the true evolution (Eq. 4.70) and let $\widehat{\psi}_{K,M}(t)$ be the output of our Alg. 4 (Eq. 4.73). When*

$$
K \geq et\,\|H\| + \log\frac{2}{\varepsilon}, \qquad M \geq \max\left(405\mathrm{Tr}(H),\ \frac{72\mathrm{Tr}(H)t}{\varepsilon}\log\frac{36\mathrm{Tr}(H)t}{\varepsilon\delta}\right),
$$

$$
\tag{4.74}
$$

then the following holds with probability $1 - \delta$,

$$\|\psi(t) - \widehat{\psi}_{K,M}(t)\| \leq \varepsilon.$$

Note that with the result above, we have that $\widehat{\psi}_{K,M}(t)$ in Eq. (4.73) (Alg. 4) approximates $\psi(t)$, with error at most ε and with probability at least $1 - \delta$, requiring a computational cost that is $O\left(\frac{st^2 \text{Tr}(H)^2}{\varepsilon^2} \log^2 \frac{1}{\delta}\right)$ in time and $O\left(\frac{t^2 \text{Tr}(H)^2}{\varepsilon^2} \log^2 \frac{1}{\delta}\right)$ in memory.

In the following we now prove the first main result of this work. To prove Theorem 10 we decompose the error into multiple contributions. Lemma 13 performs a basic decomposition of the error in terms of the distance between H and the approximation $\widehat{H}$ as well as in terms of the approximation g_K with respect to g. Lemma 14 then provides an analytic bound on the distance between H and $\widehat{H}$, expressed in terms of the expectation of eigenvalues or related matrices which are then concentrated in Lemma 15.

Lemma 13. *Let* $K, M \in \mathbb{N}$ *and* $t > 0$, *then*

$$\|\psi(t) - \widehat{\psi}_{K,M}(t)\| \leq t\left\|H - \widehat{H}\right\| + \frac{(t\left\|\widehat{H}\right\|)^{K+1}}{(K+1)!}.$$

Proof. By definition we have that $e^{ixt} = 1 + g(x)x$ with $g(x) = \sum_{k \geq 1} x^{k-1}(it)^k/k!$ and g_K is the truncated version of g. By adding and subtracting $e^{i\widehat{H}t}$, we have

$$\left\|e^{iHt}\psi - (I + g_K(\widehat{H})\widehat{H})\psi\right\| \leq \|\psi\| \left(\left\|e^{iHt} - e^{i\widehat{H}t}\right\| + \left\|e^{i\widehat{H}t} - (I + g_K(\widehat{H})\widehat{H})\right\|\right). \tag{4.75}$$

By [130],

$$\left\|e^{iHt} - e^{i\widehat{H}t}\right\| \leq t\left\|H - \widehat{H}\right\|, \tag{4.76}$$

moreover, by [131], and since $\widehat{H}$ is Hermitian and hence all the eigenvalues are real,

we have

$$\left\| e^{i\widehat{H}t} - (I + g_K(\widehat{H})\widehat{H}) \right\| \leq \frac{(t\left\|\widehat{H}\right\|)^{K+1}}{(K+1)!} \sup_{l \in [0,1]} \left\| i^{K+1} e^{il\widehat{H}t} \right\| \leq \frac{(t\left\|\widehat{H}\right\|)^{K+1}}{(K+1)!}. \quad (4.77)$$

Finally note that $\|\psi\| = 1$. $\qquad\square$

To study the norm $\left\| H - \widehat{H} \right\|$ note that, since H is positive semidefinite, there exists an operator S such that $H = SS^H$, so $H_{i,j} = s_i^H s_j$ with s_i, s_j the i-th and j-th row of S. Denote with C and $\widetilde{C}$ the operators

$$C = S^H S, \quad \widetilde{C} = \frac{1}{M} \sum_{j=1}^{M} \frac{\mathrm{Tr}(H)}{h_{t_j}} s_{t_j} s_{t_j}^H.$$

We then obtain the following result.

Lemma 14. *The following holds with probability* 1. *For any* $\tau > 0$,

$$\left\| H - \widehat{H} \right\| \leq \frac{\tau}{1 - \beta(\tau)}, \qquad \beta(\tau) = \lambda_{\max}((C + \tau I)^{-1/2}(C - \widetilde{C})(C + \tau I)^{-1/2}),$$
$$(4.78)$$

moreover $\left\| \widehat{H} \right\| \leq \|H\|$.

Proof. Define the selection matrix $V \in \mathbb{C}^{M \times 2^n}$, that is always zero except for one element in each row which is $V_{j,t_j} = 1$ for $1 \leq j \leq M$. Then we have that

$$A = HV^H, \quad B = VHV^H,$$

i.e., A is again given by the rows according to the sampled indiced $t_1, \ldots, t_M$ and B is the submatrix obtained from taking the rows and columns according to the same indices. In particular by denoting with $\widehat{P}$ the operator $\widehat{P} = S^H V^H (VSS^H V^H)^+ VS$, and recalling that $H = SS^H$ and $C = S^H S$, we have

$$\widehat{H} = AB^+ A^H = SS^H V^H (VSS^H V^H)^+ VSS^H = S\widehat{P}S^H.$$

By definition $\widehat{P}$ is an orthogonal projection operator, indeed it is symmetric and, by

definition $Q^+QQ^+ = Q^+$, for any matrix Q, then

$$\widehat{P}^2 = S^H V^H [(VSS^H V^H)^+ (VSS^H V^H)(VSS^H V^H)^+] VS = S^H V^H (VSS^H V^H)^+ VS = \widehat{P}.$$
(4.79)

Indeed this is a projection in the row space of the matrix $R := S^H V^H$, since with the singular value decomposition $R := U_R \Sigma_R V_R^H$ we have

$$\hat{P} = R^H (R^H R)^+ R = V_R \Sigma_R U_R^H U_R \Sigma_R^{-2} U_R^H U_R \Sigma_R V_R^H = V_R V_R^H,$$
(4.80)

which spans the same space as R. Finally, since $(I - \widehat{P}) = (I - \widehat{P})^2$, and $\left\| Z^H Z \right\| = \|Z\|^2$, we have

$$\left\| H - \widehat{H} \right\| = \left\| S(I - \widehat{P})S^H \right\| = \left\| S(I - \widehat{P})^2 S^H \right\| = \left\| (I - \widehat{P})S^H \right\|^2.$$
(4.81)

Note that $\widetilde{C}$ can be rewritten as $\widetilde{C} = S^H V^H LVS$, with L a diagonal matrix, with $L_{jj} = \frac{\mathrm{Tr}(H)}{Mh_{t_j}}$. Moreover t_j is sampled from the probability $p(q) = h_q/\mathrm{Tr}(H)$, so $h_{t_j} > 0$ with probability 1, then L has a finite and strictly positive diagonal, so $\widetilde{C}$ has the same range of $\widehat{P}$. Now, with $C = S^H S$, we are able to apply Proposition 3 and Proposition 7 of [55], and obtain

$$\left\| (I - \widehat{P})S^H \right\|^2 \leq \frac{\tau}{1 - \beta(\tau)}, \qquad \beta(\tau) = \lambda_{\max}((C + \tau I)^{-1/2}(C - \widetilde{C})(C + \tau I)^{-1/2}).$$
(4.82)

Finally, note that, since $\widehat{P}$ is a projection operator we have that $\left\| \widehat{P} \right\| = 1$, so

$$\left\| \widehat{H} \right\| = \left\| S\widehat{P}S^H \right\| \leq \left\| \widehat{P} \right\| \|S\|^2 \leq \|S\|^2 = \|H\|,$$

where the last step is due to the fact that $H = SS^H$. $\qquad \square$

Lemma 15. *Let $\delta \in (0,1]$ and $\tau > 0$. When*

$$M \geq \max\left(405\mathrm{Tr}(H), 67\mathrm{Tr}(H)\log\frac{\mathrm{Tr}(H)}{2\delta}\right), \qquad \tau = \frac{9\mathrm{Tr}(H)}{M}\log\frac{M}{2\delta}, \quad (4.83)$$

then with probability $1 - \delta$ *it holds that*

$$\lambda_{\max}((C + \tau I)^{-1/2}(C - \widetilde{C})(C + \tau I)^{-1/2}) \leq \frac{1}{2}.$$

Proof. Define the random variable $\zeta_j = \sqrt{\frac{\mathrm{Tr}(H)}{h_{t_j}}} s_{t_j}$, for $1 \leq j \leq M$. Note that

$$\|\zeta_j\| \leq \sqrt{\frac{\mathrm{Tr}(H)}{h_{t_j}}} \|s_{t_j}\| \leq \sqrt{\mathrm{Tr}(H)},$$

almost surely. Moreover,

$$\mathbb{E}\zeta_j\zeta_j^H = \sum_{q=1}^{2^n} p(q)\frac{\mathrm{Tr}(H)}{h_q} s_q s_q^H = \sum_{q=1}^{2^n} s_q s_q^H = S^H S = C. \tag{4.84}$$

By definition of ζ_j, we have

$$\widetilde{C} = \frac{1}{M}\sum_{j=1}^{M} \zeta_j\zeta_j^H.$$

Since ζ_j are independent for $1 \leq j \leq M$, uniformly bounded, with expectation equal to C, and with $\zeta_j^H(C + \tau I)^{-1}\zeta_j \leq \|\zeta_j\|^2 \tau^{-1} \leq \mathrm{Tr}(H)\tau^{-1}$, we can apply Proposition 8 of [55], that uses non-commutative Bernstein inequality for linear operators [132], and obtain

$$\lambda_{\max}((C + \tau I)^{-1/2}(C - \widetilde{C})(C + \tau I)^{-1/2}) \leq \frac{2\alpha}{3M} + \sqrt{\frac{2\alpha}{Mt}}, \tag{4.85}$$

with probability at least $1 - \delta$, with $\alpha = \log\frac{4\mathrm{Tr}(C)}{\tau\delta}$. Since

$$\mathrm{Tr}(C) = \mathrm{Tr}(S^H S) = \mathrm{Tr}(SS^H) = \mathrm{Tr}(H),$$

by Remark 1 of [55], we have that

$$\lambda_{\max}((C + \tau I)^{-1/2}(C - \widetilde{C})(C + \tau I)^{-1/2}) \leq \frac{1}{2},$$

with probability $1 - \delta$, when $M \geq \max(405\kappa^2, 67\kappa^2 \log \frac{\kappa^2}{2\delta})$ and τ satisfies $\frac{9\kappa^2}{M} \log \frac{M}{2\delta} \leq \tau \leq \|C\|$ (note that $\|C\| = \|H\|$), where κ^2 is a bound for the following quantity

$$
\begin{aligned}
&\inf_{\tau > 0}[(\|C\| + \tau)(\operatorname{ess\,sup} \zeta_j^H (C + \tau I)^{-1} \zeta_j)] \\
&\leq \operatorname{Tr}(H) \inf_{\tau > 0} \frac{\|H\| + \tau}{\tau} \leq \operatorname{Tr}(H) := \kappa^2,
\end{aligned}
\tag{4.86}
$$

where $\operatorname{ess\,sup}$ here denotes the essential supremum. $\qquad\square$

Now we are ready to prove Theorem 10.

Proof of Theorem 10. By Lemma 13, we have

$$
\left\| e^{iHt}\psi - (I + g_K(\widehat{H})\widehat{H})\psi \right\| \leq t \left\| H - \widehat{H} \right\| + \frac{(t \left\| \widehat{H} \right\|)^{K+1}}{(K+1)!}.
$$

Let $\tau > 0$. By Lemma 14, we know that $\left\| \widehat{H} \right\| \leq \|H\|$ and that

$$
\left\| H - \widehat{H} \right\| \leq \frac{\tau}{1 - \beta(\tau)},
$$

$$
\beta(\tau) = \lambda_{\max}((C + \tau I)^{-1/2}(C - \widetilde{C})(C + \tau I)^{-1/2}),
$$

with probability 1. Finally by Lemma 15, we have that the following holds with probability $1 - \delta$,

$$
\lambda_{\max}((C + \tau I)^{-1/2}(C - \widetilde{C})(C + \tau I)^{-1/2}) \leq \frac{1}{2},
$$

when

$$
M \geq \max\left(405 \operatorname{Tr}(H),\ 67 \operatorname{Tr}(H) \log \frac{\operatorname{Tr}(H)}{2\delta}\right)
$$

and

$$
\tau = \frac{9 \operatorname{Tr}(H)}{M} \log \frac{\operatorname{Tr}(H)}{2\delta}.
$$

So we have

$$\left\| e^{iHt}\psi - (I + g_K(\widehat{H})\widehat{H})\psi \right\| \leq \frac{18\mathrm{Tr}(H)t}{M}\log\frac{M}{2\delta} + \frac{(t\|H\|)^{K+1}}{(K+1)!},$$

with probability $1 - \delta$.

Now we select K such that $\frac{(t\|H\|)^{K+1}}{(K+1)!} \leq \frac{\varepsilon}{2}$. Since, by the Stirling approximation, we have

$$(K+1)! \geq \sqrt{2\pi}(K+1)^{K+3/2}e^{-K-1} \geq (K+1)^{K+1}e^{-K-1}.$$

Since

$$(1+x)\log(1/(1+x)) \leq -x,$$

for $x > 0$ we can select $K = et\|H\| + \log\frac{2}{\varepsilon} - 1$, such that we have

$$\begin{aligned}
\log\left(\frac{(t\|H\|)^{K+1}}{(K+1)!}\right) &\leq (K+1)\log\frac{et\|H\|}{K+1} \\
&\leq et\|H\|\left(1 + \frac{\log\frac{2}{\varepsilon}}{et\|H\|}\right)\log\frac{1}{1 + \frac{\log\frac{2}{\varepsilon}}{et\|H\|}} \\
&\leq \log\frac{\varepsilon}{2}.
\end{aligned}$$

Finally we require M, such that

$$\frac{18\mathrm{Tr}(H)t}{M}\log\frac{M}{2\delta} \leq \frac{\varepsilon}{2},$$

and select

$$M = \frac{72\mathrm{Tr}(H)t}{\varepsilon}\log\frac{36\mathrm{Tr}(H)t}{\varepsilon\delta}.$$

Then we have that

$$\frac{18\mathrm{Tr}(H)t}{M}\log\frac{M}{2\delta} \leq \frac{\varepsilon}{2}\frac{\log\frac{36\mathrm{Tr}(H)t}{\varepsilon\delta} + \log\log\frac{36\mathrm{Tr}(H)t}{\varepsilon\delta}}{2\log\frac{36\mathrm{Tr}(H)t}{\varepsilon\delta}} \leq \frac{\varepsilon}{2}.$$

$\square$

Next, we generalise this results to arbitrary Hermitian matrices under the assumption that these are row-searchable, i.e. assuming the ability to sample according to some leverage of the rows. This will lead to our second result for classical Hamiltonian simulation with the Nyström method.

Algorithm for row-searchable Hermitian matrices As mentioned, in this section we now generalise our previous result and derive an algorithm for simulating arbitrary Hermitian matrices. We again provide guarantees on the runtime and errors of the algorithm, generally under the assumption that H is row-searchable. Our algorithm for the general case has slightly worse guarantees compared to the SPD case, which is to be expected. Let in the following again s be the maximum number of non-zero elements in any of the rows of H, ε be the error in the approximation of the output states of the algorithm w.r.t. the ideal $\psi(t)$, and t the evolution time of the simulation. Let further K be the order of the truncated series expansions and M the number of samples we take for the approximation.

In the following we again start by describing the algorithm, and then derive bounds on the runtime and error.

For arbitrary matrices H we will use the following algorithm. Sample $M \in \mathbb{N}$ independent indices $t_1, \ldots t_M$, with probability $p(i) = \frac{\|h_i\|^2}{\|H\|_F^2}$, $1 \leq i \leq 2^n$, where h_i is the i-th row of H (sample via Alg. 3). Let $A \in \mathbb{C}^{2^n \times M}$ be the matrix defined by

$$A = \left[\frac{1}{\sqrt{Mp(t_1)}} h_{t_1}, \ldots, \frac{1}{\sqrt{Mp(t_1)}} h_{t_M} \right].$$

We then approximate H via the matrix $\hat{H}^2 = AA^H$.

Next, we again define two functions that we will use to approximate the exponential e^{ix},

$$f(x) = \frac{\cos(\sqrt{x}) - 1}{x}, \quad g(x) = \frac{\sin(\sqrt{x}) - \sqrt{x}}{x\sqrt{x}},$$

and denote with f_K and g_K the K-truncated Taylor expansions of f and g, for $K \in \mathbb{N}$, i.e.,

$$f_K(x) = \sum_{j=0}^{K} \frac{(-1)^{j+1} x^j}{(2j+2)!}, \quad g_K(x) = \sum_{j=0}^{K} \frac{(-1)^{j+1} x^j}{(2j+3)!}.$$

In particular note that

$$e^{ix} = 1 + ix + f(x^2)x^2 + ig(x^2)x^3.$$

Similar in spirit to the previous approach for SPD H, we hence approximate e^{ix} via the functions f_K and g_K. The final approximation is then given by

$$\widehat{\psi}_{K,M}(t) = \psi + itu + t^2 A f_K(t^2 A^H A)v + it^3 A g_K(t^2 A^H A)z, \qquad (4.87)$$

where $u = \hat{H}\psi$, $v = A^H \psi$, $z = A^H u$. The products $f_k(A^H A)v$ and $A g_K(A^H A)z$ are done by again exploiting the Taylor series form of the two functions and performing only matrix vector products similar to Alg. 4. Recall that s is the maximum number of non-zero elements in the rows of H, and q the number of non-zero elements in ψ. The algorithm then requires $O(sq)$ in space and time to compute u, $O(M \min(s, q))$ in time and $O(M)$ in space to compute v and $O(Ms)$ in time and space to compute z. We therefore obtain a total computational complexity of

$$\text{time}: \ O(sq + M\min(s,q) + sMK), \qquad (4.88)$$

$$\text{space}: \ O(s(q+M)). \qquad (4.89)$$

Note that if $s > M$ is it possible to further reduce the memory requirements at the cost of more computational time, by computing $B = A^H A$, which can be done in blocks and require $O(sM^2)$ in time and $O(M^2)$ in memory, and then compute

$$\widehat{\psi}_{K,M}(t) = \psi + itu + t^2 A f_K(t^2 B)v + it^3 A g_K(t^2 B)z.$$

In this case the computational cost would be

$$\text{time}: \ O\left(sq + M\min(s,q) + M^2(s+K)\right), \qquad (4.90)$$

$$\text{space}: \ O\left(sq + M^2\right). \qquad (4.91)$$

The properties of the this algorithm are summarised in the following theorem, which

is a formal statement of Theorem 8:

Theorem 11 (Algorithm for simulating row-samplable Hermitian matrices). *Let $\delta, \varepsilon \in (0,1]$. Let $t > 0$ and $K, M \in \mathbb{N}$, where K is the number of terms in the truncated series expansions of $g(\widehat{H})$ and M the number of samples we take for the approximation, and let $t > 0$. Let $\psi(t)$ be the true evolution (Eq. 4.70) and let $\widehat{\psi}_{K,M}(t)$ be computed as in Eq. 4.87. When*

$$M \geq \frac{256 t^4 (1 + t^2 \|H\|^2) \|H\|_F^2 \|H\|^2}{\varepsilon^2} \log \frac{4 \|H\|_F^2}{\delta \|H\|^2}, \tag{4.92}$$

$$K \geq 4t \sqrt{\|H\|^2 + \varepsilon} + \log \frac{4(1 + t\|H\|)}{\varepsilon}, \tag{4.93}$$

then

$$\|\widehat{\psi}_{K,M}(t) - \psi(t)\| \leq \varepsilon,$$

with probability at least $1 - \delta$.

Note that with the result above, we have that $\widehat{\psi}_{K,M}(t)$ in Eq. (4.87) approximates $\psi(t)$, with error at most ε and with probability at least $1 - \delta$, requiring a computational cost that is $O\left(sq + M \min(s,q) + M^2(s+K)\right)$ in time and $O\left(sq + M^2\right)$ is memory.

Combining Eq. 4.90 and 4.91 with Eq. 4.92 and 4.93, the whole computational complexity of the algorithm described in this section, is

$$\text{time}: \ O\left(sq + \frac{t^9 \|H\|_F^4 \|H\|^7}{\varepsilon^4} \left(n + \log \frac{1}{\delta}\right)^2\right), \tag{4.94}$$

$$\text{space}: \ O\left(sq + \frac{t^8 \|H\|_F^4 \|H\|^6}{\varepsilon^4} \left(n + \log \frac{1}{\delta}\right)^2\right), \tag{4.95}$$

where the quantity $\log \frac{4\|H\|_F^2}{\delta \|H\|^2}$ in Eq. 4.92 was bounded using the following inequality

$$\log \frac{\|H\|_F^2}{\|H\|^2} \leq \log \frac{2^n \lambda_{MAX}^2}{\lambda_{MAX}^2} = n,$$

where λ_{MAX} is the biggest eigenvalue of H.

Observe now that simulation of the time evolution of αI does only change the phase of the time evolution, where $I \in \mathbb{C}^{N \times N}$ is the identity matrix and α some real parameter. We can hence perform the time evolution of $\tilde{H} := H - \alpha I$, since for any efficient classical description of the input state we can apply the time evolution of the diagonal matrix $e^{-i\alpha It}$. We can then optimise the parameter α such that the Frobenius norm of the operator $\tilde{H}$ is minimized, i.e.

$$\alpha = \operatorname*{argmin}_{\alpha} \left\| \tilde{H} \right\|_F^2 = \operatorname*{argmin}_{\alpha} \left\| H - \alpha I \right\|_F^2, \tag{4.96}$$

from which we obtain the condition $\alpha = \frac{\mathrm{Tr}(H)}{2^n}$. Since, in order for the algorithm to be efficient, we require that $\left\| \tilde{H} \right\|_F$ is bounded by polylog$N$. Using the spectral theorem, and the fact that the Frobenius norm is unitarily invariant, this in turn gives us after a bit of algebra the condition

$$\|H\|_F^2 - \frac{1}{N}\mathrm{Tr}(H)^2 \leq O\left(\mathrm{polylog}(N)\right), \tag{4.97}$$

for which we can simulate the Hamiltonian H efficiently.

We now prove the second main result of this work and establish the correctness of the above results.

Proof of Theorem 11. Denote with

$$\widehat{Z}_K(Ht, At) = I + itH + t^2 A f_K(t^2 A^H A) A^H + it^3 A g_K(t^2 A^H A) A^H H,$$

$$\widehat{Z}(Ht, At) = I + itH + t^2 A f(t^2 A^H A) A^H + it^3 A g(t^2 A^H A) A^H H.$$

By definition of $\widehat{\psi}_{K,M}(t)$ and the fact that $\|\psi\| = 1$, we have

$$\|\widehat{\psi}_{K,M}(t) - \psi(t)\| \leq \left\| \widehat{Z}_K(At, Ht) - e^{iHt} \right\| \|\psi\|$$

$$\leq \left\| \widehat{Z}_K(At, Ht) - \widehat{Z}(Ht, At) \right\| + \left\| \widehat{Z}(Ht, At) - e^{iHt} \right\|.$$

We first study $\left\| \widehat{Z}(Ht, At) - e^{iHt} \right\|$. Define $l(x) = f(x)x$ and $m(x) = g(x)x$. Note that,

by the spectral theorem, we have

$$\widehat{Z}(Ht, At) = I + itH + t^2 Af(t^2 A^H A)A^H + it^3 Ag(t^2 A^H A)A^H H$$

$$= I + itH + t^2 f(t^2 AA^H)AA^H + it^3 g(t^2 AA^H)AA^H$$

$$= I + itH + l(t^2 AA^H) + itm(t^2 AA^H)H.$$

Since

$$e^{ixt} = 1 + ixt + l(t^2 x^2) + itm(t^2 x^2)x,$$

we have

$$\left\| \widehat{Z}(Ht, At) - e^{iHt} \right\| = \left\| l(t^2 AA^H) - l(t^2 H^2) + itm(t^2 AA^H)H - itm(t^2 H^2)H \right\|$$

$$\leq \left\| l(t^2 AA^H) - l(t^2 H^2) \right\| + t \left\| m(t^2 AA^H) - m(t^2 H^2) \right\| \|H\|.$$

To bound the norms in l, m we will apply Thm. 1.4.1 of [133]. The theorem state that if a function $f \in L^\infty(\mathbb{R})$, i.e. f is in the function space which elements are the essentially bounded measurable functions, it is entirely on $\mathbb{C}$ and satisfies $|f(z)| \leq e^{\sigma|z|}$ for any $z \in \mathbb{C}$. Then $\|f(A) - f(B)\| \leq \sigma \|f\|_{L^\infty(\mathbb{R})} \|A - B\|$. Note that

$$|l(z)| = \left| \sum_{j=1}^\infty \frac{(-1)^j z^j}{(2j)!} \right| \leq \sum_{j=1}^\infty \frac{|z|^j}{(2j)!} \leq \sum_{j=1}^\infty \frac{|z|^j}{j!} \leq e^{|z|},$$

$$|l(z)| = \left| \sum_{j=1}^\infty \frac{(-1)^j z^j}{(2j+1)!} \right| \leq \sum_{j=1}^\infty \frac{|z|^j}{(2j+1)!} \leq \sum_{j=1}^\infty \frac{|z|^j}{j!} \leq e^{|z|}.$$

Moreover it is easy to see that $\|l\|_{L^\infty(\mathbb{R})}, \|m\|_{L^\infty(\mathbb{R})} \leq 2$. So

$$\left\| \widehat{Z}(Ht, At) - e^{iHt} \right\| \leq 2(1 + t\|H\|) \left\| t^2 AA^H - t^2 H^2 \right\|$$

$$= 2t^2(1 + t\|H\|) \left\| AA^H - H^2 \right\|.$$

Now note that, by defining the random variable $\zeta_i = \frac{1}{p(t_i)} h_{t_i} h_{t_i}^H$, we have that

$$AA^H = \frac{1}{M} \sum_{i=1}^{M} \zeta_i,$$

$$\mathbb{E}[\zeta_i] = \sum_{q=1}^{2^n} p(q) \frac{1}{p(q)} h_{t_i} h_{t_i}^H = H^2, \quad \forall i.$$

Let $\tau > 0$. By applying Thm. 1 of [134] (or Prop. 9 in [55]), for which

$$\left\| AA^H - H^2 \right\| \leq \sqrt{\frac{\|H\|_F^2 \|H\|^2 \tau}{M}} + \frac{2 \|H\|_F^2 \|H\|^2 \tau}{M},$$

with probability at least $1 - 4 \frac{\|H\|_F^2}{\|H\|^2} \tau / (e^\tau - \tau - 1)$. Now since

$$1 - 4 \frac{\|H\|_F^2}{\|H\|^2} \tau / (e^\tau - \tau - 1) \geq 1 - e^\tau,$$

when $\tau \geq e$, by selecting

$$\tau = 2 \log \frac{4 \|H\|_F^2}{\|H\| \delta},$$

we have that the equation above holds with probability at least $1 - \delta$.

Let $\eta > 0$, by selecting

$$M = 4 \eta^{-2} \|H\|_F^2 \|H\|^2 \tau,$$

we then obtain

$$\left\| AA^H - H^2 \right\| \leq \eta,$$

with probability at least $1 - \delta$.

Now we study $\left\| \widehat{Z}_K(At, Ht) - \widehat{Z}(Ht, At) \right\|$. Denote with a, b the functions $a(x) = l(x^2)$, $b(x) = m(x^2)$ and with a_K, b_K the associated K-truncated Taylor expansions. Note that $a(x) = \cos(x) - 1$, while $b(x) = (\sin(x) - x)/x$. Now by definition

of $\widehat{Z}_K$ and $\widehat{Z}$, we have

$$\left\| \widehat{Z}_K(At, Ht) - \widehat{Z}(Ht, At) \right\| \leq \left\| a_K(t\sqrt{AA^H}) - a(t\sqrt{AA^H}) \right\|$$
$$+ t \left\| H \right\| \left\| b_K(t\sqrt{AA^H}) - b(t\sqrt{AA^H}) \right\|.$$

Note that, since

$$\sum_j x^{2j}/(2j)! = \cosh(x) \leq 2e^{|x|},$$

and

$$|(a_K - a)(x)| = \left| \sum_{j=K+2} (-1)^j \frac{x^{2j}}{(2j)!} \right|$$
$$\leq \frac{|x|^{2K+4}}{(2K+4)!} \sum_{j=0} \frac{|x|^{2j}}{(2j)!} \frac{(2K+4)!(2j)!}{(2j+2K+4)!}$$
$$\leq \frac{2|x|^{2K+4}e^{|x|}}{(2K+4)!},$$
$$|(b_K - b)(x)| = \left| \sum_{j=K+2} (-1)^j \frac{x^{2j}}{(2j+1)!} \right|$$
$$\leq \frac{|x|^{2K+4}}{(2K+4)!} \sum_{j=0} \frac{|x|^{2j}}{(2j)!} \frac{(2K+4)!(2j)!}{(2j+2K+5)!}$$
$$\leq \frac{2|x|^{2K+4}e^{|x|}}{(2K+4)!}.$$

Let $R > 0, \beta \in (0,1]$. Now note that, by Stirling approximation, $c! \geq e^{c\log\frac{c}{e}}$, so by

selecting $K = \frac{e^2}{2}R + \log(\frac{1}{\beta})$, we have for any $|x| \leq R$,

$$\log \left| \frac{2|x|^{2K+4}e^{|x|}}{(2K+4)!} \right| \leq |x| + (2K+4)\log \frac{e|x|}{2K+4}$$

$$\leq R + (2K+4)\log \frac{eR}{2K+4}$$

$$\leq R - \left(e^2 R + 4 + \log \frac{1}{\beta} \right) \log \left(e + \frac{4 + \log 1\beta}{eR} \right)$$

$$\leq R - \left(e^2 R + 4 + \log \frac{1}{\beta} \right)$$

$$\leq -(e^2 - 1)R - 4 - \log \frac{1}{\beta}$$

$$\leq -\log \frac{1}{\beta}.$$

So, by choosing $K \geq \log \frac{1}{\beta} + e^2 R/2$, we have $|a_K(x) - a(x)|, |b_K(x) - b(x)| \leq \beta$. With this we finally obtain

$$\left\| \widehat{Z}_K(At, Ht) - \widehat{Z}(Ht, At) \right\| \leq 2(1 + t\|H\|)\beta,$$

when $K \geq e^2 t \|A\| /2 + \log \frac{1}{\beta}$, and therefore we have

$$\| \widehat{\psi}_{K,M}(t) - \psi(t) \| \leq 2t^2(1 + t\|H\|)\eta + 2(1 + t\|H\|)\beta,$$

with probability at least $1 - \delta$, when

$$M \geq 8\eta^{-2} \|H\|_F^2 \|H\|^2 \log \frac{4\|H\|_F^2}{\|H\|^2 \delta}, \quad K \geq 4t\|A\| + \log \frac{1}{\beta}.$$

In particular, by choosing $\eta = \varepsilon/(4t^2(1 + t\|H\|))$ and $\beta = \varepsilon/(4(1 + t\|H\|))$, we have

$$\| \widehat{\psi}_{K,M}(t) - \psi(t) \| \leq \varepsilon,$$

with probability at least $1 - \delta$.

With this result in mind note that, in the event where $\left\| AA^H - H^2 \right\| \leq \varepsilon$, we

have that

$$| \left\| AA^H \right\| - \left\| H \right\|^2 | \leq \left\| AA^H - H^2 \right\| \leq \varepsilon,$$

and therefore, $\|A\| \leq \sqrt{\|H\|^2 + \varepsilon}$. □

4.4.6 Beyond the Nyström method

While the Nyström method has been applied very successful over the past decade or so, in recent years, it also has been improved upon using better approximations for symmetric matrices. The results of [135] indicated that the Nyström method has advantages over the random feature method [136], the closest competitor, both theoretically and empirically. However, as it has recently been established, even the Nyström method does not attain high accuracy in general. A model which improves the accuracy of the Nyström method, is the so-called prototype model [137, 138]. The prototype model performs first a random sketch on the input matrix H, i.e., $C = HS$, where S is a sketching or sampling matrix which samples M rows of H, and then computes the intersection matrix U^* as

$$U^* := \text{argmin}_U \left\| H - CUC^H \right\|_F^2 = C^+ H (C^+)^H \in \mathbb{R}^{M \times M}.$$

The model then approximates H by CU^*C^H. While the intial versions of the prototype model were not efficient due to the cost of calculating U^*, [139] improved these results by approximately calculating the optimal U^*, which led to a higher accuracy compared to the Nyström method, but also to an improved runtime. As future work, we leave it open, whether these methods can be used to obtain improved classical algorithms for Hamiltonian simulation and quantum machine learning in general.

4.4.7 Conclusion

As we mentioned in the introduction of the chapter, our results are closely related to the so-called 'quantum-inspired' or 'dequantisation' results, which followed the work by Tang [17]. It is obvious that our *row-searchable* condition is fundamentally equivalent to the *sample and query* access requirement, and both our and other results indicate that the state preparation condition requires a careful assessment in

order to determine whether a quantum algorithm provides an actual advantage over its classical equivalent. In particular, our sampling scheme for Hermitian matrices is equivalent to the sampling scheme based on the classical memory structure which we introduced earlier, although for PSD matrices we use a more computationally efficient variant. While we also provide a method based on binary-trees to compute the sampling probabilities, the tree structure of the memory immediately allows us to execute such a sampling process in practice. The main difference is therefore that we use a traditional memory structure and provide a fast way to calculate the marginals using a binary tree (see Sec. 4.4.5), while [17] assumes a memory structure, which allows one to sample efficiently according to this distribution.

Notably, we believe our results can be further improved by using the rejection sampling methods as is done in all quantum-inspired approaches. The rejection sampling is required in order to achieve faster algorithms as they allow us to sample from the output distribution rather than putting out the entirety of it – as is the case with our algorithm.

Based on our results for Hamiltonian simulation, and supported by the large amount of recent quantum-inspired results for QML applications, we believe it less likely that QML algorithms will admit any exponential speedups compared to classical algorithms. The main reason is because, unlike their classical counterparts, most QML algorithms require strong input assumptions. As we have seen, these caveats arise due to two reasons. First, the fast loading of large input data into a quantum computer is a very powerful assumption, which indeed leads to similarly powerful classical algorithms. Second, extracting the results from an output quantum state is hard in general. Tang's [17] breakthrough result in 2018 for the quantum recommendation systems algorithm, which was previously believed to be one of the strongest candidates for a practical exponential speedup in QML, indeed implied that the quantum algorithm does not give an exponential speedup. Tang's algorithm solves the same problem as the quantum algorithm and just incurs a polynomial slow-down. To do so, it relies on similar methods to the ones we described for Hamiltonian simulation with the Nyström method, and combines these with

classical rejection sampling.

While exponential speedups therefore appear not to be possible in general, in specific cases, such as problems with sparsity assumptions, these dequantisation techniques cannot be applied and QML algorithms still might offer the long-sought exponential advantage.

The most well known quantum algorithm of this resistant type is the HHL algorithm for sparse matrix inversion [36], which indeed is BQP-complete. Although HHL has a range of caveats, see e.g. [43], perhaps it can be useful in certain instances. Works that build on top of the HHL, such as Zhao et al. on Gaussian process regression [140] and Lloyd et al. on topological data analysis [141] might indeed overcome these drawbacks and achieve a super-polynomial quantum speedup.

The question whether such quantum or quantum-inspired algorithms will be useful in practice is, however, even harder to answer. Indeed, for algorithms with a theoretical polynomial speedup an actual advantage can only be established through proper benchmarks and performance analysis such as a scaling analysis. The recent work by Arrazola et al. [142], for example, implemented and tested the quantum-inspired algorithms for regression and recommendation systems and compared these against existing state-of-the-art implementations. While giving various insights, however, it is unclear whether the analysis in question is valid. In order to assure that the algorithms are tested properly, more modern sketching algorithms would need to be used, since the ones that we and also other authors rely on do not necessarily provide performance close to the current state-of-the-art. The reason for this is that they were chosen to obtain complexity theoretic results, i.e., to demonstrate that from a theoretical perspective the quantum algorithms do not have an exponential advantage. As was also pointed out in [21] Dahiya, Konomis, and Woodruff [143] already conducted an empirical study of sketching algorithms for low-rank approximation on both synthetic datasets and the movielens dataset, and reported that their implementation "finds a solution with cost at most 10 times the optimal one [...] but does so 10 times faster", which is in contrast to the results of Arrazola et al. [142]. Additionally, using self-implemented algorithms and compar-

ing these in a non-optimised way to highly optimised numerical packages such as LAPACK, BLAS, or others, is usually not helpful when evaluating the performance of algorithms.

We therefore conclude that quantum machine learning algorithms still might have a polynomial advantage over classical ones, but in practice many challenges exist and need to be overcome. It is hence unclear as of today whether they will be able to provide any meaningful advantage over classical machine learning algorithms. In the future we hope to see results that implement the quantum algorithms and analyse the resulting overhead, and ultimately benchmark these against their quantum-inspired counterparts. However, we believe that such benchmarks will not be possible in the near future, as the requirements on the quantum hardware are far beyond what is possible today.

Chapter 5

Promising avenues for QML

In the previous chapters, we have tried to answer the main questions of this thesis, i.e., whether (a) quantum machine learning algorithms can offer an advantage over their classical counterparts from a statistical perspective, and (b) whether randomised approaches can be used to achieve similarly powerful classical algorithms. We answered both of these questions in the context of supervised quantum machine learning algorithms for classical data, i.e., data that is stored in some form of a classical memory. However, data could also be obtained as the immediate output of a quantum process, and the QML algorithm could therefore also directly use the resulting quantum states as input that could in principle resolve qRAM-related challenges. Indeed, in this regime (QQ in Fig. 1.1), it is very likely that classical machine learning algorithms will have difficulties, since they are fundamentally not able to use the input except through prior sampling (measurement) of the state, which implies a potential loss of information. The ability to prepare, i.e., generate arbitrary input states with a polynomially sized circuit could therefore enable useful quantum machine learning algorithms.

In this chapter, we now briefly discuss a possible future avenue for QML, namely the generative learning of quantum distributions and quantum states. In particular, we provide a method for fully quantum generative training of quantum Boltzmann machines with both visible and hidden units. To do this, we rely on the quantum relative entropy as an objective function, which is significant since prior methods cannot do so due to mathematical challenges. The mathematical challenges

are a result of the gradient evaluation which is required in the training process. Our method is highly relevant, as it allows us to efficiently estimate gradients even for nearly parallel states, which has been impossible for all previous methods. We can therefore use our algorithm even to approximate cloning and state preparation for arbitrary input states. We present in the following two novel methods for solving this problem. The first method given an efficient algorithm for a class of restricted quantum Boltzmann machines with mutually commuting Hamiltonians on the hidden units. In order to train it with gradient descent and the quantum relative entropy as an objective function, we use a variational upper bound. The second one generalises the first result to generic quantum Boltzmann machines by using high-order divided difference methods and linear-combinations of unitaries to approximate the exact gradient of the relative entropy. Both methods are efficient under the assumption that Gibbs state preparation is efficient and assuming that the Hamiltonian is a sparse, row-computable matrix.

5.1   Generative quantum machine learning

One objective of QML is to design models that can learn in quantum mechanical settings [144, 5, 145, 146, 147, 148], i.e., models which are able to quickly identify patterns in data, i.e., quantum state vectors, which inhabit an exponentially large vector space [149, 150]. The sheer size of these vectors therefore restricts the computations that can be performed efficiently. As we have seen above, one of the main bottlenecks for many QML applications (e.g. supervised learning) is the input data preparation. While qRAM appears not to be a solution to the common data read-in problem, other methods for the fast preparation of quantum states might enable useful QML applications.

The clearest cases where quantum machine learning can therefore provide an advantage is when the input is already in the form of quantum data, i.e., quantum states. An alternative way to qRAM to efficiently generate quantum states according to a desired distribution are generative models. Generative models can derive concise descriptions of large quantum states [151, 152, 153, 28], and be able to

Work	Data	Algorithm	Objective	Hidden Units
Wiebe et al. [159]	Classical	Quantum (Classical model)	Maximum likelihood	Yes
Amin et al. [156]	Classical	Quantum (Quantum model)	Maximum likelihood	Yes (But cannot be trained)
Kieferova et al. [151]	Classical (Tomography)	Quantum	Maximum likelihood (KL-divergence)	Yes
Kieferova et al. [151]	Quantum	Quantum	Relative entropy	No
Our work	Quantum	Quantum (Restricted H)	Relative entropy (Variational bound)	Yes
Our work	Quantum	Quantum	Relative entropy	Yes

Figure 5.1: Comparison of previous training algorithms for quantum Boltzmann machines. The models have a varying cost function (objective), contain (are able to be trained with) hidden units, and have different input data (classical or quantum).

prepare states with in principle polynomially sized circuits.

The most general representation for the input, which is the natural analog of a quantum training set, would be a density operator ρ, which is a positive semi-definite trace-1 Hermitian matrix that, roughly speaking, describes a probability distribution over the input quantum state vectors. The goal in quantum generative training is to learn a unitary operator $V : |0\rangle \mapsto \sigma$, which takes as input a quantum state $|0\rangle$, and prepares the density matrix σ, by taking a small (polynomial) number of samples from ρ, with the condition that under some chosen distance measure D, $D(\rho, \sigma)$, is small. For example, for the L_1 norm this could be $D(\rho, \sigma) = \|\rho - \sigma\|_1$. In the quantum information community this task is known as partial tomography [151] or approximate cloning [154]. A similar task is to replicate a conditional probability distribution over a label subspace. Such approaches play a crucial role in QAOA-based quantum neural networks.

5.1.1 Related work

Various approaches have been put forward to solve the challenge of generative learning in the quantum domain [151, 153, 155, 156, 157], but to date, all proposed solutions suffer from drawbacks due to data input requirements, vanishing gradients, or an inability to learn with hidden units. In this chapter, we present two novel approaches for training quantum Boltzmann machines [151, 158, 156]. Our approaches resolve all restrictions of prior art, and therefore address a major open problem in generative quantum machine learning. Figure 5.1 summarises the key results and prior art.

5.2 Boltzmann and quantum Boltzmann machines

Boltzmann machines are a physics inspired class of neural network [160, 161, 162, 163, 164, 165, 166, 167], which have gained increasing popularity and found numerous applications over the last decade [168, 166, 169, 170]. More recently, they have been used in the generative description of complex quantum systems [171, 172, 173]. Boltzmann machines are an immediate approach when we want to perform generative learning, since they are a physically inspired model which resembles many common quantum systems. Through this similarity, they are also highly suitable to be implemented on quantum computers. To be more precise, a Boltzmann machine is defined by the energy which arises from the interactions in the physical system. As such, it prescribes an energy to every configuration of this system, and then generates samples from a distribution which assigns probabilities to the states according to the exponential of the states energy. This distribution is in classical statistical physics known as the canonical ensemble. The explicit model is given by

$$\sigma_v(H) = \text{Tr}_h\left(\frac{e^{-H}}{Z}\right) = \frac{\text{Tr}_h\, e^{-H}}{\text{Tr}\, e^{-H}}, \tag{5.1}$$

where $\text{Tr}_h(\cdot)$ is the partial trace over the so-called *Hidden* subsystem, an auxillary sub-system which allows the model to build correlations between different nodes in the *Visible* subsystem. In the case of classical Boltzmann machines, the Hamiltonian H is an energy function that assigns to each state an energy, resulting in a diagonal matrix. In the quantum mechanical case, however, we can obtain superpositions (i.e., linear combinations) of different states, and the energy function therefore turns into the Hamiltonian of the quantum Boltzmann, i.e., a general Hermitian matrix which has off-diagonal entries.

As mentioned earlier, for a quantum system the dimension of the Hamiltonian grows exponentially with the number of units such as qubits or orbitals, denoted n, of the system, i.e., $H \in \mathbb{C}^{2^n \times 2^n}$.

Generative quantum Boltzmann training is then defined as the task of finding the parameters or so-called weights θ, which parameterise the Hamiltonian $H =$

$H(\theta)$, such that

$$H = \text{argmin}_{H(\theta)}\left(\text{D}(\rho, \sigma_v(H))\right),$$

where D is again an appropriately chosen distance, or divergence, function. For example, the quantum analogue of an all-visible Boltzmann machine with n_v units would then take the form

$$H(\theta) = \sum_{n=1}^{n_v} \theta_{2n-1}\sigma_x^{(n)} + \theta_{2n}\sigma_z^{(n)} + \sum_{n>n'} \theta_{(n,n')}\sigma_z^{(n)}\sigma_z^{(n')}. \tag{5.2}$$

Here $\sigma_z^{(n)}$ and $\sigma_x^{(n)}$ are Pauli matrices acting on qubit (unit) n.

We now provide a formal definition of the quantum Boltzmann machine.

Definition 5. *A quantum Boltzmann machine to be a quantum mechanical system that acts on a tensor product of Hilbert spaces $\mathcal{H}_v \otimes \mathcal{H}_h \in \mathbb{C}^{2^n}$ that correspond to the visible and hidden subsystems of the Boltzmann machine. It further has a Hamiltonian of the form $H \in \mathbb{C}^{2^n \times 2^n}$ such that $\|H - diag(H)\| > 0$. The quantum Boltzmann machine takes these parameters and then outputs a state of the form $\text{Tr}_h\left(\frac{e^{-H}}{\text{Tr}(e^{-H})}\right).$*

For classical data we hence want to minimise the distance between two different distributions, namely the input and output distribution. The natural way to measure such a distance is the Kullback-Leibler (KL) divergence. In the case of the quantum Boltzmann machine, in contrast, we want to minimise the distance between two quantum states (density matrices), and the natural notion of distance changes to the quantum relative entropy:

$$S(\rho|\sigma_v) = \text{Tr}(\rho \log \rho) - \text{Tr}(\rho \log \sigma_v). \tag{5.3}$$

For diagonal ρ and σ_v this indeed reduces to the (classical) KL divergence, and it is zero if and only if $\rho = \sigma_v$.

For Boltzmann machines with visible units, the gradient of the relative entropy can be computed in a straightforward manner because here $\sigma_v = e^{-H}/Z$ and since $\log(e^{-H}/Z) = -H - \log(Z)$, we can easily compute the matrix derivatives. This

is however not the case in general, and no methods are known for the generative training of Boltzmann machines for the quantum relative entropy loss function, if hidden units are present. The main challenge which restricts an easy solution in this case is the evaluation of the partial trace in $\log(\mathrm{Tr}_h e^{-H}/Z)$, which prevents us from simplifying the logarithm term which is required for the gradient.

In the following, we will provide two practical methods for training quantum Boltzmann machines with both hidden and visible units. We provide two different approaches for achieving this. The first method only applies to a restricted case, however it allows us to propose a more efficient scheme that relies on a variational upper bound on the quantum relative entropy in order to calculate the derivatives. The second approach is more general method, but requires more resources. For the second approach we rely on recent techniques from quantum simulation to approximate the exact expression for the gradient. For this we rely on a Fourier series approximation and high-order divided difference formulas in place of the analytic derivative. Under the assumption that Gibbs state preparation is efficient, which we expect to hold in most practical cases, both methods are efficient.

We note however, that this assumption is indeed not valid in general, in particular, since the possibility of efficient Gibbs state preparation would imply QMA $\subseteq$ BQP which is unlikely to hold.

While the here presented results might be interesting for several readers, in order to preserve the flow and keep the presentation simple, we will include the proofs of technical lemmas in the appendix of the thesis.

5.3 Training quantum Boltzmann machines

We now present methods for training quantum Boltzmann machines.
We begin by defining the quantum relative entropy, which we use as cost function for our quantum Boltzmann machine (QBM) with hidden units. Recall that ρ is our input distribution, and $\sigma = \mathrm{Tr}_h \left[e^{-H}/\mathrm{Tr}\left(e^{-H}\right)\right]$ is the output distribution, where we perform the partial trace over the hidden units. With this, the cost function is hence

given by

$$\mathscr{O}_\rho(H) = S\left(\rho \middle| \mathrm{Tr}_h\left[e^{-H}/\mathrm{Tr}\left(e^{-H}\right)\right]\right), \tag{5.4}$$

where $S(\rho|\sigma_v)$ is the quantum relative entropy as defined in eq. 5.3. A few remarks. First, note that it is possible to add a regularisation term in order to penalise undesired quantum correlations in the model [151]. Second, since we will only derive gradient based algorithms for the training, we need to evaluate the gradient of the cost function, which requires the gradient of the quantum relative entropy.

As previously mentioned, this is possible to do in a closed-form expression for the case of an all-visible Boltzmann machine, which corresponds to $\dim(\mathscr{H}_h) = 1$. The gradient in this case takes the form

$$\frac{\partial \mathscr{O}_\rho(H)}{\partial \theta} = -\mathrm{Tr}\left(\frac{\partial}{\partial \theta}\rho \log \sigma\right), \tag{5.5}$$

which can be simplified using $\log(\exp(-H)) = -H$ and Duhamels formula to obtain the following equation for the gradient, denoting with $\partial_\theta := \partial/\partial\theta$,

$$\mathrm{Tr}\left(\rho\partial_\theta H\right) - \mathrm{Tr}\left(e^{-H}\partial_\theta H\right)/\mathrm{Tr}\left(e^{-H}\right). \tag{5.6}$$

However, this gradient formula is does not hold any longer if we include hidden units. If we hence want to include hidden units, then we need to additionally trace out the subsystem. Doing so results in the majorised distribution from eq. 5.1, which also changes the cost function into the form described in eq. 5.4. Note that $H = H(\theta)$ is depending on the training weights. We will adjust these in the training process with the goal to match σ to ρ – the target density matrix. Omitting the $\mathrm{Tr}\left(\rho\log\rho\right)$ since it is a constant, we therefore obtain

$$\frac{\partial \mathscr{O}_\rho(H)}{\partial \theta} = -\mathrm{Tr}\left(\frac{\partial}{\partial \theta}\rho \log \sigma_v\right), \tag{5.7}$$

for the gradient of the objective function.

In the following sections, we now present our two approaches for evaluating the gradient in eq. 5.7. The first one is less general, but gives us an easy imple-

mentable algorithm with strong bounds. The second one, on the hand, can be applied to arbitrary problem instances, and therefore presents a general purpose gradient based algorithm for training quantum Boltzmann machines with the relative entropy objective. The results are to be expected, since the no-free-lunch theorem suggests that no good bounds can be obtained without assumptions on the problem instance. Indeed, our general algorithm can in principle result in an exponentially worse complexity compared to the specialised one. However, we are expecting that for most practical applications this will not be the case, and our algorithm will in hence be applicable for most cases of practical interest. To our knowledge, we hence present the first algorithm of this kind, which is able to train arbitrary arbitrary quantum Boltzmann machines efficiently.

5.3.1 Variational training for restricted Hamiltonians

Our first approach optimises a variational upper bound of the objective function, for a restricted but still practical setting. This approach results in a fast and easy way to implement quantum algorithms which, however, is less general due to the input assumptions we are making. These assumptions are required in order to overcome challenges related the evaluation of matrix functions, for which classical calculus fails.

Concretely, we express the Hamiltonian in this case as

$$H = H_v + H_h + H_{\text{int}}, \tag{5.8}$$

i.e., a decomposition of the Hamiltonian into a part acting on the visible layers, the hidden layers and a third interaction Hamiltonian that creates correlations between the two. In particular, we further assume for simplicity that there are two sets of operators $\{v_k\}$ and $\{h_k\}$ composed of $D = W_v + W_h + W_{\text{int}}$ terms such that

$$H_v = \sum_{k=1}^{W_v} \theta_k v_k \otimes I, \qquad\qquad H_h = \sum_{k=W_v+1}^{W_v+W_h} \theta_k I \otimes h_k$$

$$H_{\text{int}} = \sum_{k=W_v+W_h+1}^{W_v+W_h+W_{\text{int}}} \theta_k v_k \otimes h_k, \qquad [h_k, h_j] = 0 \ \forall \ j, k, \tag{5.9}$$

which implies that the Hamiltonian can in general be expressed as

$$H = \sum_{k=1}^{D} \theta_k v_k \otimes h_k. \tag{5.10}$$

We break up the Hamiltonian into this form to emphasise the qualitative difference between the types of terms that can appear in this model. Note that we generally assume throughout this article that v_k, h_k are unitary operators, which is indeed given in most common instances.

By choosing this particular form of the Hamiltonian in (5.9), we want to force the non-commuting terms, i.e., terms for which it holds that the commutator $[v_k, h_k] \neq 0$, to act only on the visible units of the model. On the other hand, only commuting Hamiltonian terms act on the hidden register, and we can therefore express the eigenvalues and eigenvectors for the Hamiltonian as

$$H |v_h\rangle \otimes |h\rangle = \lambda_{v_h,h} |v_h\rangle \otimes |h\rangle. \tag{5.11}$$

Note that here both the conditional eigenvectors and eigenvalues for the visible subsystem are functions of the eigenvector $|h\rangle$ in the hidden register, and we hence denote these as $v_h, \lambda_{v_h,h}$ respectively. This allows the hidden units to select between eigenbases to interpret the input data while also penalising portions of the accessible Hilbert space that are not supported by the training data. However, since the hidden units commute they cannot be used to construct a non-diagonal eigenbasis. While this division between the visible and hidden layers on the one hand helps us to build an intuition about the model, on the other hand – more importantly – it allows us to derive a more efficient training algorithms that exploit this fact.

As previously mentioned, for the first result we rely on a variational bound of the objective function in order to train the quantum Boltzmann machine weights for a Hamiltonian H of the form given in (5.10). We can express this variational bound compactly in terms of a thermal expectation against a fictitious thermal probability distribution. We define this expectation below.

Definition 6. *Let $\tilde{H}_h = \sum_k \theta_k \text{Tr}(\rho v_k) h_k$ be the Hamiltonian acting conditioned*

on the visible subspace only on the hidden subsystem of the Hamiltonian $H :=$
$\sum_k \theta_k v_k \otimes h_k$. *Then we define the expectation value over the marginal distribution*
over the hidden variables h as

$$\mathbb{E}_h(\cdot) = \sum_h \frac{(\cdot)e^{-\mathrm{Tr}(\rho \tilde{H}_h)}}{\sum_h e^{-\mathrm{Tr}(\rho \tilde{H}_h)}}. \tag{5.12}$$

Using this we derive an upper bound on S in section B.1 of the Appendix,
which leads to the following lemma.

Lemma 16. *Assume that the Hamiltonian H of the quantum Boltzmann machine*
takes the form described in eq. 5.10, where θ_k are the parameters which deter-
mine the interaction strength and v_k, h_k are unitary operators. Furthermore, let
$h_k |h\rangle = E_{h,k} |h\rangle$ *be the eigenvalues of the hidden subsystem, and $\mathbb{E}_h(\cdot)$ as given by*
Definition 6, i.e., the expectation value over the effective Boltzmann distribution of
the visible layer with $\tilde{H}_h := \sum_k E_{h,k} \theta_k v_k$. Then, a variational upper bound $\widetilde{S}$ of the
objective function, meaning that $\widetilde{S}(\rho|H) \geq S(\rho|e^{-H}/Z)$, is given by

$$\widetilde{S}(\rho|H) := \mathrm{Tr}(\rho \log \rho) + \mathrm{Tr}\left(\rho \sum_k \mathbb{E}_h\left[E_{h,k} \theta_k v_k\right] + \mathbb{E}_h\left[\log \alpha_h\right]\right) + \log Z, \tag{5.13}$$

where

$$\alpha_h = \frac{e^{-\mathrm{Tr}(\rho \tilde{H}_h)}}{\sum_h e^{-\mathrm{Tr}(\rho \tilde{H}_h)}}$$

is the corresponding Gibbs distribution for the visible units.

The proof that (5.13) is a variational bound proceeds in two steps. First, we
note that for any probability distribution α_h

$$\mathrm{Tr}\left(\rho \log \left(\sum_{h=1}^N e^{-\sum_k E_{h,k} \theta_k v_k}\right)\right) = \mathrm{Tr}\left(\rho \log \left(\sum_{h=1}^N \alpha_h \frac{e^{-\sum_k E_{h,k} \theta_k v_k}/\alpha_h}{\sum_{h'} \alpha_{h'}}\right)\right) \tag{5.14}$$

We then apply Jensen's inequality and minimise the result over all α_h. This not only
verifies that $\widetilde{S}(\rho|H) \geq S(\rho|H)$ but also yields a variational bound. The details of
the proof can be found in Equation B.5 in section B.1 of the appendix.

Using the above assumptions we derive the gradient of the variational upper

bound of the relative entropy in the Section B.2 of the Appendix. We summarise the result in Lemma 17.

Lemma 17. *Assume that the Hamiltonian H of the quantum Boltzmann machine takes the form described in eq. 5.10, where θ_k are the parameters which determine the interaction strength and v_k, h_k are unitary operators. Furthermore, let $h_k |h\rangle = E_{h,k} |h\rangle$ be the eigenvalues of the hidden subsystem, and $\mathbb{E}_h(\cdot)$ as given by Definition 6, i.e., the expectation value over the effective Boltzmann distribution of the visible layer with $\tilde{H}_h := \sum_k E_{h,k} \theta_k v_k$. Then, the derivatives of $\widetilde{S}$ with respect to the parameters of the Boltzmann machine are given by*

$$\frac{\partial \widetilde{S}(\rho|H)}{\partial \theta_p} = \mathbb{E}_h \left[\mathrm{Tr} \left(\rho E_{h,p} v_p \right) \right] - \mathrm{Tr} \left(\frac{\partial H}{\partial \theta_p} \frac{e^{-H}}{Z} \right). \tag{5.15}$$

As a sanity check of our results, we can consider the case of no interactions between the visible and the hidden layer. Doing so, we observe that the gradient above reduces to the case of the visible Boltzmann machine, which was treated in [151], resulting in the gradient

$$\mathrm{Tr} \left(\rho \partial_{\theta_p} H \right) - \mathrm{Tr} \left(\frac{e^{-H}}{Z} \partial_{\theta_p} H \right), \tag{5.16}$$

under our assumption on the form of H, $\partial_{\theta_p} H = v_p$.

From Lemma 17, we know the form of the derivatives of the relative entropy w.r.t. any parameter θ_p via Eq. 5.15. To understand the complexity of evaluating this gradient, we approach each term separately. The second term can easily be evaluated by preparing the Gibbs state $\sigma_{Gibbs} := e^{-H}/Z$, and then evaluating the expectation value of the operator $\partial_{\theta_j} H$ w.r.t. this Gibbs state. In practice we can do so using amplitude estimation for the Hadamard test [174], which is a standard procedure and we describe it in algorithm 8 in section A.2 of the appendix. The computational complexity of this procedure is easy to evaluate. If T_{Gibbs} is the query complexity for the Gibbs state preparation, the query complexity of the whole algorithm including the phase estimation step is then given by $O(T_{Gibbs}/\varepsilon)$ for an

ε-accurate estimate of phase estimation.

Next, we derive an algorithm to evaluate the first term, which requires a more involved process. For this, note first that we can evaluate each term $\mathrm{Tr}(\rho v_k)$ independently from $\mathbb{E}_h\left[E_{h,p}\right]$, and individually for all $k \in [D]$, i.e., all D dimensions of the gradient. This can be done via the Hadamard test for v_k which we recapitulate in section A.2 of the appendix, assuming v_k is unitary. More generally, for non-unitary v_k we could evaluate this term using a linear combination of unitary operations.

Therefore, the remaining task is to evaluate the terms $\mathbb{E}_h\left[E_{h,p}\right]$ in (5.15), which reduces to sampling elements according to the distribution $\{\alpha_h\}$, recalling that h_p applied to the subsystem has eigenvalues $E_{h,p}$. For this we need to be able to create a Gibbs distribution for the effective Hamiltonian $\tilde{H}_h = \sum_k \theta_k \mathrm{Tr}(\rho v_k) h_k$ which contains only D terms and can hence be evaluated efficiently as long as D is small, which we can generally assume to be true. In order to sample according to the distribution $\{\alpha_h\}$, we first evaluate the factors $\theta_k \mathrm{Tr}(\rho v_k)$ in the sum over k via the Hadamard test, and then use these in order to implement the Gibbs distribution $\exp(-\tilde{H}_h)/\tilde{Z}$ for the Hamiltonian

$$\tilde{H}_h = \sum_k \theta_k \mathrm{Tr}(\rho v_k) h_k.$$

The algorithm is summarised in Algorithm 5.

The algorithm is build on three main subroutines. The first one is Gibbs state preparation, which is a known routine which we recapitulate in Theorem 15 in the appendix. The two remaining routines are the Hadamard test and amplitude estimation, both are well established quantum algorithms. The Hadamard test, will allow us to estimate the probability of the outcome. This is concretely given by

$$\Pr(0) = \frac{1}{2}\left(1 + \mathrm{Re}\left\langle \psi\right|_{Gibbs}(h_k \otimes I)\left|\psi\right\rangle_{Gibbs}\right) = \frac{1}{2}\left(1 + \sum_h \frac{e^{-E_h}E_{h,k}}{Z}\right), \quad (5.18)$$

i.e., from $\Pr(0)$ we can easily infer the estimate of $\mathbb{E}_h\left[E_{h,k}\right]$ up to precision ε for all the k terms, since the last part is equivalent to $\frac{1}{2}\left(1 + \mathbb{E}_h[E_{h,k}]\right)$. To speed up the time for the evaluation of the probability $\Pr(0)$, we use amplitude estimation. We

---

Algorithm 5 Variational gradient estimation - term 1

Input: An upper bound $\tilde{S}(\rho|H)$ on the quantum relative entropy, density matrix $\rho \in \mathbb{C}^{2^n \times 2^n}$, and Hamiltonian $H \in \mathbb{C}^{2^n \times 2^n}$.

Output: Estimate $\mathscr{S}$ of the gradient $\nabla \tilde{S}$ which fulfills Thm. 12.

1. Use Gibbs state preparation to create the Gibbs distribution for the effective Hamiltonian $\tilde{H}_h = \sum_k \theta_k \text{Tr}(\rho v_k) h_k$ with sparsity d.

2. Prepare a Hadamard test state, i.e., prepare an ancilla qubit in the $|+\rangle$-state and apply a controlled-h_k conditioned on the ancilla register, followed by a Hadamard gate, i.e.,

$$|\phi\rangle := \frac{1}{2}\left(|0\rangle\left(|\psi\rangle_{Gibbs} + (h_k \otimes I)|\psi\rangle_{Gibbs}\right) + |1\rangle\left(|\psi\rangle_{Gibbs} - (h_k \otimes I)|\psi\rangle_{Gibbs}\right)\right)$$

$$(5.17)$$

where $|\psi\rangle_{Gibbs} := \sum_h \frac{e^{-E_h/2}}{\sqrt{Z}}|h\rangle_A |\phi_h\rangle_B$ is the purified Gibbs state.

3. Perform amplitude estimation on the $|0\rangle$ state, we need to implement the amplitude estimation with reflector $P := -2|0\rangle\langle 0| + I$, and operator $G := (2|\phi\rangle\langle\phi| - I)(P \otimes I)$.

4. Measure now the phase estimation register which returns an $\tilde{\varepsilon}$-estimate of the probability $\frac{1}{2}\left(1 + \mathbb{E}_h[E_{h,k}]\right)$ of the Hadamard test to return 0

6. Repeat the procedure for all D terms and output the first term of $\nabla \tilde{S}$.

---

recapitulate this procedure in detail in the supplemental material in section A.1. In this case, we let $P := -2|0\rangle\langle 0| + I$ be the reflector, where I is the identity which is just the Pauli z matrix up to a global phase, and let $G := (2|\phi\rangle\langle\phi| - I)(P \otimes I)$, for $|\phi\rangle$ being the state after the Hadamard test prior to the measurement. The operator G has then the eigenvalue $\mu_\pm = \pm e^{\pm i2\theta}$, where $2\theta = \arcsin\sqrt{\text{Pr}(0)}$, and $\text{Pr}(0)$ is the probability to measure the ancilla qubit in the $|0\rangle$ state. Let now T_{Gibbs} be the query complexity for preparing the purified Gibbs state (c.f. eq (B.17) in the appendix). We can then perform phase estimation with precision ε for the operator G requiring $O(T_{Gibbs}/\tilde{\varepsilon})$ queries to the oracle of H.

In section B.2.1 of the appendix we analyse the runtime and error of the above algorithm. The result is summarised in Theorem 12.

Theorem 12. *Assume that the Hamiltonian H of the quantum Boltzmann machine takes the form described in eq. 5.10, where θ_k are the parameters which determine the interaction strength and v_k, h_k are unitary operators. Furthermore, let $h_k|h\rangle = E_{h,k}|h\rangle$ be the eigenvalues of the hidden subsystem, and $\mathbb{E}_h(\cdot)$ as given by*

Definition 6, i.e., the expectation value over the effective Boltzmann distribution of the visible layer with $\tilde{H}_h := \sum_k E_{h,k} \theta_k v_k$, and suppose that $I \preceq \tilde{H}_h$ with bounded spectral norm $\left\| \tilde{H}_h(\theta) \right\| \leq \|\theta\|_1$, and let $\tilde{H}_h$ be d-sparse. Then $\mathscr{S} \in \mathbb{R}^D$ can be computed for any $\varepsilon \in (0, \max\{1/3, 4 \max_{h,p} |E_{h,p}|\})$ such that

$$\left\| \mathscr{S} - \nabla \tilde{S} \right\|_{max} \leq \varepsilon, \tag{5.19}$$

with

$$\widetilde{\mathscr{O}} \left(\sqrt{\xi} \frac{D \|\theta\|_1 \, dn^2}{\varepsilon} \right), \tag{5.20}$$

queries to the oracle O_H and O_ρ with probability at least 2/3, where $\|\theta\|_1$ is the sum of absolute values of the parameters of the Hamiltonian, $\xi := \max[N/z, N_h/z_h]$, $N = 2^n$, $N_h = 2^{n_h}$, and z, z_h are known lower bounds on the partition functions for the Gibbs state of H and $\tilde{H}_h$ respectively.

Theorem 12 shows that the computational complexity of estimating the gradient grows the closer we get to a pure state, since for a pure state the inverse temperature $\beta \to \infty$, and therefore the norm $\|H(\theta)\| \to \infty$, as the Hamiltonian is depending on the parameters, and hence the type of state we describe. In such cases we typically would rely on alternative techniques. However, this cannot be generically improved because otherwise we would be able to find minimum energy configurations using a number of queries in $o(\sqrt{N})$, which would violate lower bounds for Grover's search. Therefore more precise statements of the complexity will require further restrictions on the classes of problem Hamiltonians to avoid lower bounds imposed by Grover's search and similar algorithms.

5.3.2 Gradient based training for general Hamiltonians

While our first scheme for training quantum Boltzmann machines is only applicable in case of a restricted Hamiltonian, the second scheme, which we will present next, holds for arbitrary Hamiltonians. In order to calculate the gradient, we use higher order divided difference estimates for the relative entropy objective based on function approximation schemes. For this we generate differentiation formulas by differentiating an interpolant. The main ideas are simple: First we construct an

interpolating polynomial from the data. Second, an approximation of the derivative at any point is obtained via a direct differentiation of the interpolant.

Concretely we perform the following steps.

We first approximate the logarithm via a Fourier-like approximation, i.e., $\log \sigma_v \to \log_{K,M} \sigma_v$, where the subscripts K, M indicate the level of truncation similar to [175]. This will yield a Fourier-like series in terms of σ_v, i.e., $\sum_m c_m \exp(im\pi\sigma_v)$.

Next, we need to evaluate the gradient of the function

$$\mathrm{Tr}\left(\frac{\partial}{\partial \theta}\rho \log_{K,M}(\sigma_v)\right).$$

Taking the derivative yields many terms of the form

$$\int_0^1 ds e^{(ism\pi\sigma_v)}\frac{\partial \sigma_v}{\partial \theta}e^{(i(1-s)m\pi\sigma_v)}, \tag{5.21}$$

as a result of the Duhamel's formula for the derivative of exponentials of operators (c.f., Sec. 2 in the mathematical preliminaries). Each term in this expansion can furthermore be evaluated separately via a sampling procedure, since the terms in Eq. 5.21 can be approximated by

$$\mathbb{E}_s\left[e^{(ism\pi\sigma_v)}\frac{\partial \sigma_v}{\partial \theta}e^{(i(1-s)m\pi\sigma_v)}\right].$$

Furthermore, since we only have a logarithmic number of terms, we can combine the results of the individual terms via classical postprocessing once we have evaluated the trace.

Now, we apply a divided difference scheme to approximate the gradient term $\frac{\partial \sigma_v}{\partial \theta}$ which results in an interpolation polynomial $\mathcal{L}_{\mu,j}$ of order l (for l being the number of points at which we evaluate the function) in σ_v which we can efficiently evaluate.

However, evaluating these terms is still not trivial. The final step consists hence of implementing a routine which allows us to evaluate these terms on a quantum

---

Algorithm 6 Gradient estimation via series approximations

Input: Density matrices $\rho \in \mathbb{C}^{2^n \times 2^n}$ and $\sigma_v \in \mathbb{C}^{2^{n_v} \times 2^{n_v}}$, precalculated parameters K, M and Fourier-like series for the gradient as described in eq. B.57.
Output: Estimate $\mathscr{G}$ of the gradient $\nabla_\theta \text{Tr}(\rho \log \sigma_v)$ with guarantees in Thm. 12.

1. Prepare the $|+\rangle \otimes \rho$ state for the Hadamard test.
2. Conditionally on the first qubit apply sample based Hamiltonian simulation to ρ, i.e., for $U := e^{\frac{is\pi m}{2}\sigma_v} e^{\frac{i\pi m'}{2}\sigma_v(\theta_j)} e^{\frac{i(1-s)\pi m}{2}\sigma_v}$, apply $|0\rangle\langle 0| \otimes I + |1\rangle 1 \otimes U$.
3. Apply another Hadamard gate to the first qubit.
4. Repeat the above procedure and measure the final state each time and return the averaged output.

---

device. In order to do so, we once again make use of the Fourier series approach. This time we take the simple idea of approximating the density operator σ_v by the series of itself, i.e., $\sigma_v \approx F(\sigma_v) := \sum_{m'} c_{m'} \exp(im\pi m' \sigma_v)$, which we can implement conveniently via sample based Hamiltonian simulation [13, 57].

Following these steps we obtain the expression in Eq. B.57. The real part of

$$\sum_{m=-M_1}^{M_1} \sum_{m'=-M_2}^{M_2} \frac{ic_m \tilde{c}_{m'} m\pi}{2} \sum_{j=0}^{\mu} \mathscr{L}'_{\mu,j}(\theta) \mathbb{E}_{s\in[0,1]} \left[\text{Tr}\left(\rho e^{\frac{is\pi m}{2}\sigma_v} e^{\frac{i\pi m'}{2}\sigma_v(\theta_j)} e^{\frac{i(1-s)\pi m}{2}\sigma_v} \right) \right].$$

$$(5.22)$$

then approximates $\partial_\theta \text{Tr}(\rho \log \sigma_v)$ with at most ε error, where $\mathscr{L}'_{\mu,j}$ is the derivative of the interpolation polynomial which we obtain using divided differences, and $\{c_i\}_i, \{\tilde{c}_j\}_j$ are coefficients of the approximation polynomials, which can efficiently be evaluated classically. We can evaluate each term in the sum separately and combine the results then via classical post-processing, i.e., by using the quantum computer to evaluate terms containing the trace.

The main challenge for the algorithmic evaluation hence to compute the terms

$$\text{Tr}\left(\rho e^{\frac{is\pi m}{2}\sigma_v} e^{\frac{i\pi m'}{2}\sigma_v(\theta_j)} e^{\frac{i(1-s)\pi m}{2}\sigma_v} \right). \qquad (5.23)$$

Evaluating this expression is done through Algorithm 6, relies on two established subroutines, namely sample based Hamiltonian simulation [13, 57], and the Hadamard test which we discussed earlier. Note that the sample based Hamiltonian simulation approach introduces an additional ε_h-error in trace norm, which we also

need to take into account in the analysis. In section B.3.1 of the appendix we derive the following guarantees for Algorithm. 6.

Theorem 13. *Let ρ, σ_v being two density matrices, $\|\sigma_v\| < 1/\pi$, and we have access to an oracle O_H that computes the locations of non-zero matrix elements in each row and their values for the d-sparse Hamiltonian $H(\theta)$ (as per [80]) and an oracle O_ρ which returns copies of purified density matrix of the data ρ, and $\varepsilon \in (0, 1/6)$ an error parameter. With probability at least $2/3$ we can obtain an estimate $\mathscr{G}$ of the gradient w.r.t. $\theta \in \mathbb{R}^D$ of the relative entropy $\nabla_\theta \text{Tr}(\rho \log \sigma_v)$ such that*

$$\|\nabla_\theta \text{Tr}(\rho \log \sigma_v) - \mathscr{G}\|_{max} \leq \varepsilon, \tag{5.24}$$

with

$$\tilde{O}\left(\sqrt{\frac{N}{z}} \frac{D\|H(\theta)\| d\mu^5 \gamma}{\varepsilon^3}\right), \tag{5.25}$$

queries to O_H and O_ρ, where $\mu \in O(n_h + \log(1/\varepsilon))$, $\|\partial_\theta \sigma_v\| \leq e^\gamma$, $\|\sigma_v\| \geq 2^{-n_v}$ for n_v being the number of visible units and n_h being the number of hidden units, and

$$\tilde{O}\left(poly\left(\gamma, n_v, n_h, \log(1/\varepsilon)\right)\right)$$

classical precomputation.

In order to obtain the bounds in Theorem 13, we decompose the total error into the errors that we incur at each step of the approximation scheme,

$$\left|\partial_\theta \text{Tr}(\rho \log \sigma_v) - \partial_\theta \text{Tr}\left(\rho \log^s_{K_1, M_1} \tilde{\sigma}_v\right)\right| \leq \sum_i \sigma_i(\rho) \cdot \left\|\partial_\theta[\log \sigma_v - \log^s_{K_1, M_1} \tilde{\sigma}_v]\right\|$$

$$\leq \sum_i \sigma_i(\rho) \cdot \left(\left\|\partial_\theta[\log \sigma_v - \log_{K_1, M_1} \sigma_v]\right\|\right.$$

$$\left. + \left\|\partial_\theta[\log_{K_1, M_1} \sigma_v - \log_{K_1, M_1} \tilde{\sigma}_v]\right\| + \left\|\partial_\theta[\log_{K_1, M_1} \tilde{\sigma}_v - \log^s_{K_1, M_1} \tilde{\sigma}_v]\right\|\right). \tag{5.26}$$

Then bounding each term separately and adjusting the parameters to obtain an overall error of ε allows us to obtain the above result. We are hence able to use this

procedure to efficiently obtain gradient estimates for a QBM with hidden units, while making minimal assumptions on the input data.

5.4 Conclusion

Generative models may play a crucial role in making quantum machine learning algorithms practical, as they yield concise models for complex quantum states that have no known a priori structure. In this chapter, we solved an outstanding problem in the field of generative quantum modelling: We overcame the previous inability to train quantum generative models with a quantum relative entropy objective for a QBM with hidden units. In particular, the inability to train models with hidden units was a substantial drawback.

Our results show that, given an efficient subroutine for preparing Gibbs states and an efficient algorithm for computing the matrix elements of the Hamiltonian, one can efficiently train a quantum Boltzmann machine with the quantum relative entropy.

While a number of problems remain with our algorithms, namely (1) that we have not given a lower bound for the query complexity, and hence cannot answer the question whether a linear scaling in $\|\theta\|_1$ is optimal, (2) that our results are only efficient if the complexity of performing the Gibbs state preparation is low, we still show a first efficient algorithm for training such generative quantum models. This is important, as such models might in the future open up avenues for fast state preparation, and could therefore overcome some of the drawbacks related to the data access, since the here presented model could also be trained in an online manner, i.e., whenever a data point arrives it could be used to evaluate and update the gradient. Once the model is trained, we could then use it to generate states from the input distribution and use these in a larger QML routine. With further optimisation, it is our hope that quantum Boltzmann machines may not be just a theoretical tool that can one day be used to model quantum states, but that they can be used to show an early use of quantum computers for modelling quantum or classical data sets in near term quantum hardware.

The here discussed results for learning with quantum data lead also to a more philosophical question, which was raised to us by Iordanis Kerenidis: Is the learning of quantum distributions from polynomially-sized circuits truly a quantum problem, or is it rather classical machine learning?

In the following we now briefly attempt to answer this question as well.

Much of the attention of the QML literature for learning of quantum data has been focused on learning a circuit that can generate a given input quantum distribution. However, such distributions may be assumed to be derived from a polynomially-sized quantum circuit, which implies that the data has an efficient (polynomially-sized) classical description, namely the quantum circuit that generated it (or the circuits).

Under the assumption that there exist polynomially-sized quantum circuits, which generate the quantum data that we aim to learn, the problem can, however, be seen as a classical machine learning problem. This is because the learning can happen entirely with classical means by learning the gates of the circuit with the slight addition that one can run the quantum circuits as well. Note that running the quantum circuit can be seen as the query of a cost function or a sample generation, depending on the context. In this context, we indeed do not anticipate that a differentiation can be made between classical or quantum data and hence the learning.

The difference might still occur when we are dealing with quantum states that are derived from inaccessible (black-box) quantum processes. In this case, we cannot guarantee that there exists a quantum circuit that has a polynomially sized circuit, which is able to generate the data. However, we are still able to learn an approximation to these states, see e.g. [176]. In this scenario, we can describe the problem again through a two-step process as a classical machine learning problem. First, we can for each quantum state from the data set learn a circuit that approximates the input state. Then, we take the set of circuits as an input model to the classical machine learning routine, and learn a classification of these. In this sense, the problem is entirely classical, with the difference that we rely on a quantum circuit for the evaluation of the cost function (e.g., the distance of the quantum states).

Given this, we anticipate that for certain tasks, such as classification tasks, no exponential advantage should be possible for quantum data.

Chapter 6

Conclusions

In this thesis, we have investigated the area of quantum machine learning, the combination of machine learning and quantum mechanics (in general), and in particular here the use of quantum computers to perform machine learning. We have particularly investigated supervised quantum machine learning methods, and answered two important questions:

The first research question 1 asked what is the performance of supervised quantum machine learning algorithms under the common assumptions of statistical learning theory. Our analysis indicated that most of the established and existing quantum machine learning algorithms, which are all of a theoretical nature, do not offer any advantage over their classical counterparts under common assumptions. In particular, we gave the first results that rigorously study the performance of quantum machine learning algorithms in light of the target generalisation accuracy of the learning algorithm, which is of a statistical nature. We show that any quantum machine learning algorithm will exhibit a polynomial scaling in the number of training samples – immediately excluding any exponential speedup. A more in-depth analysis of the algorithms demonstrated that indeed, taking this into account, many of the quantum machine learning algorithms do not offer any advantage over their classical counterparts.

This insight also allows us to immediately understand that noise in current generations of quantum computers significantly limits our ability to obtain good solutions, since it would immediately introduce an additional linear term in error

decomposition of Eq. 3.22.

A range of questions crucially remain open. First, while we better understand the limitations of quantum algorithms due to the generalisation ability, we are unable to investigate these bounds with respect to the complexity of the Hypothesis space. A core open question and possible avenue forward to better understand the discrepancy between classical and quantum methods is the study of the complexity $c(\mathscr{H})$, i.e., the measure of the complexity of $\mathscr{H}$ (such as the VC dimension, covering numbers, or the Rademacher complexity [47, 44]). This might indeed clarify whether quantum kernels or other approaches could be used to improve the performance of quantum algorithms. First attempts in this direction exist in the works of [177] and [178]. In particular, the latter identified with the tools of statistical learning theory under which circumstances a quantum kernel can give an advantage. While these results are very encouraging, [178] also demonstrates that for most problems a quantum advantage for learning problems is highly unlikely. In particular, the questions remain for what type of practical problems such a quantum kernel would give an advantage.

Another open question of this thesis is the effect of the condition number. In practice, conditioning is a crucial aspect of many algorithms, and while we give here an indication about possible outcomes, a clear quantum advantage would need to take the condition number also into account. In particular, since existing algorithms, such as the one by Clader et al. [11], have been shown not to work [12], new approaches are needed to overcome the drawbacks of current quantum algorithms compared to their classical counterparts.

The second research question 2 on the other hand investigated the challenge posed by the oracles that are used to establish exponentially faster (theoretical) computational complexities, i.e., run times. Most supervised quantum machine learning methods assume the existence of a logarithmic state preparation process, i.e., a quantum equivalent of a random access memory called qRAM [6]. We inspected this integral part of quantum machine learning under the lens of randomised numerical linear algebra, and asked what is the comparative advantage of quantum

algorithms over classical ones if we can efficiently sample from the data in both cases.

We established on the example of Hamiltonian simulation, that given the ability to sample efficiently from the input data – according to some form of leverage – we are able to obtain computational complexities for Hamiltonian simulation that are independent of the dimension of the input matrix. To do this, we first derived a quantum algorithm for Hamiltonian simulation, which is based on a quantum random memory. Next, we developed a classical algorithm that performed a similar though harder task, but which also did not have any explicit dependency on the dimension. Our results were some of the first that introduced randomised methods into the classical simulation of quantum dynamics and one of the first to study the comparative advantage of classical randomised algorithms over quantum ones.

While we didn't manage to extend our algorithms immediately to any application in quantum machine learning, shortly after we published our results, Ewin Tang derived the now famous quantum-inspired algorithm for recommendation systems [17], which uses randomised methods similar to the ones we introduced for Hamiltonian simulation in the context of quantum machine learning. We hence also put our results in context by adding a discussion regarding the literature of quantum-inspired algorithms.

Notably, the larger question remains if such memory structures such as qRAM will ever be practically feasible. Older results [74] indicate that even small errors can diminish a quantum advantage for search, and therefore could also crucially affect quantum machine learning algortihms. While recent research [75] claims that such errors could be sustained by quantum machine learning algorithms, we believe this question is far from clear today.

A further constraint arises from the actual costs – end-to-end – to run a quantum algorithm using such a memory structure. In reality, the cost to build and run such devices still need to be significantly reduced in order to make any advantage economic.

In the final chapter of the thesis, we then turned to future avenues for quan-

tum machine learning. Since we established that the state preparation is one of the biggest bottlenecks for successful applications and a possible advantage of quantum machine learning over classical methods, we hence investigate generative quantum models. We believe that these might hold the key to making quantum machine learning models useful in the intermediate future by replacing potentially infeasible quantum memory models such as qRAM. We resolve a major open question in the area of generative quantum modelling. Namely, we derive efficient algorithms for the gradient based training of Quantum Boltzmann Machines with hidden layers using the quantum relative entropy as an objective function. We present in particular two algorithms. One that applies in a more restricted case and is more efficient. And a second one that applies to arbitrary instances but is less efficient. Our results hold under the assumption that we can efficiently prepare Gibbs distributions, which is not true in general, but to be expected to hold for most practical applications.

A key question that is not treated in this thesis is how can we establish good benchmarks for new quantum algorithms such as in quantum machine learning. This slightly extends the more general questions on how to judge normal optimisation algorithms for exactly the statistical guarantees that we aim to achieve.

In practice, it might be required to perform the following steps to evaluate a new algorithm:

1. Obtain the theoretical time complexity of an algorithm (asymptotic).

2. Determine the dependencies (error, input dimensions, condition number, etc.) that are in common with the classical algorithm and identify how these compare.

3. Use known bounds on the target generalisation accuracy and see if the advantage remains.

4. Estimate overheads (pre-factors) for the algorithm using the best known results to go from the asymptotic scaling to a concrete cost.

5. Use estimates of condition number and other properties for some common

machine learning datasets to see if a practical advantage is feasible (since the overhead).

6. Take other error sources into account and check if the advantage still remains (i.e., take account error rates in the device and how they affect the bounds). This must also include overheads from error correction.

Benchmarks as the above one, which are of a theoretical nature, are the only option to establish a quantum advantage unless fully error-corrected devices are available for testing. However, defining a common way to perform such benchmarks remains an open question in the quantum algorithms community.

More recently, the community started to investigate the area of so-called quantum kernel methods. These might enable us to find new ways to leverage quantum computing to better identify patterns in certain types of data.

From a practical point of view, however, the ultimate test for any method is the demonstrated performance on a benchmark data set, such as MNIST. While today's quantum computers appear to be far away from this, it will be the only true way to establish a performance advantage for quantum machine learning over classical methods.

One interesting aspect that we have not considered in this thesis is the reconciliation of the research questions one and two, namely how classical randomised algorithms and quantum algortihms fit into the picture of statistical learning theory. It appears that a much fairer comparison is possible between quantum and classical randomised methods. Indeed, there exist classical algorithms for machine learning which leverage randomisation, optimal rates (generalisation accuracy), and other aspects such as preconditioning to obtain performance improvements. Such methods typically lead to a better time-scaling while also resulting in optimal learning rates - take for example FALCON [10].

Whether quantum machine learning will become useful in the future using generative models for the data access, or whether the polynomial speedup of quantum machine learning algorithms over their classical (quantum-inspired) versions will hold remains an open question.

Ultimately, we believe that these questions can only truly be verified through implementation and benchmarking. Due to the current state of quantum computing hardware, we however believe that such benchmarks will unlikely be possible in the next years, and that researchers will need to continue to make theoretical progress long before we will be able to resolve the questions of a 'quantum advantage' through an actual benchmark.

Algorithm 7 Amplitude estimation

Input: Density matrix ρ, unitary operator $U : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$, qubit registers $|0\rangle \otimes |0\rangle^{\otimes n}$.

Output: An $\tilde{\varepsilon}$ close estimate of $\mathrm{Tr}(U\rho)$.

1. Initialize two registers of appropriate sizes to the state $|0\rangle \mathscr{A} |0\rangle$, where $\mathscr{A}$ is a unitary transformation which prepares the input state, i.e., $|\psi\rangle = \mathscr{A}|0\rangle$.

2. Apply the quantum Fourier transform $\mathrm{QFT}_N : |x\rangle \to \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i x y / N} |y\rangle$ for $0 \le x < N$, to the first register.

3. Apply $\Lambda_N(Q)$ to the second register, i.e., let $\Lambda_N(U) : |j\rangle |y\rangle \to |j\rangle (U^j |y\rangle)$ for $0 \le j < N$, then we apply $\Lambda_N(Q)$ where $Q := -\mathscr{A} S_0 \mathscr{A}^\dagger S_t$ is the Grover's operator.

4. Apply $\mathrm{QFT}_N^\dagger$ to the first register.

5. Return $\tilde{a} = \sin^2(\pi \frac{\tilde{\theta}}{N})$.

Appendix A

Appendix 1: Quantum Subroutines

A.1 Amplitude estimation

In the following we describe the established *amplitude estimation algorithm* [179]:

Algorithm 7 describes the amplitude estimation algorithm. The output is an ε-close estimate of the target amplitude. Note that in step (3), S_0 changes the sign of the amplitude if and only if the state is the zero state $|0\rangle$, and S_t is the sign-flip operator for the target state, i.e., if $|x\rangle$ is the desired outcome, then $S_t := I - 2|x\rangle \langle x|$.

The algorithm can be summarized as the unitary transformation

$$\left((\mathrm{QFT}^\dagger \otimes I) \Lambda_N(Q) (\mathrm{QFT}_N \otimes I) \right)$$

Algorithm 8 Variational gradient estimation - term 2

Input: Density matrix ρ, unitary operator $U : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$, qubit registers $|0\rangle \otimes |0\rangle^{\otimes n}$.
Output: An $\tilde{\varepsilon}$ close estimate of $\text{Tr}(U\rho)$.
1. Prepare the first qubits $|+\rangle$ state and initialize the second register to 0.
2. Use an appropriate subroutine to prepare the density matrix ρ on the second register to obtain the state $|+\rangle\langle+| \otimes \rho$.
3. Apply a controlled operation $|0\rangle\langle0| \otimes I_{2^n} + |1\rangle\langle1| \otimes U$, followed by a Hadamard gate.
4. Perform amplitude estimation on the $|0\rangle$ state, via the reflector $P := -2|0\rangle\langle0| + I$, and operator $G := (2\rho - I)(P \otimes I)$.
5. Measure now the phase estimation register which returns an $\tilde{\varepsilon}$-estimate of the probability $\frac{1}{2}(1 + \text{Re}[\text{Tr}(U\rho)])$ of the Hadamard test to return 0.
6. Repeat the procedure for an additional controled application of $\exp(i\pi/2)$ in step (3) to recover also the imaginary part of the result.
7. Return the real and imaginary part of the probability estimates.

---

applied to the state $|0\rangle \mathscr{A} |0\rangle$, followed by a measurement of the first register and classical post-processing returns an estimate $\tilde{\theta}$ of the amplitude of the desired outcome such that $|\theta - \tilde{\theta}| \leq \varepsilon$ with probability at least $8/\pi^2$. The result is summarized in the following theorem, which states a slightly more general version.

Theorem 14 (Amplitude Estimation [179])**.** *For any positive integer k, the Amplitude Estimation Algorithm returns an estimate $\tilde{a}$ ($0 \leq \tilde{a} \leq 1$) such that*

$$|\tilde{a} - a| \leq 2\pi k \frac{\sqrt{a(1-a)}}{N} + k^2 \frac{\pi^2}{N^2}$$

with probability at least $\frac{8}{\pi^2} \approx 0.81$ for $k = 1$ and with probability greater than $1 - \frac{1}{2(k-1)}$ for $k \geq 2$. If $a = 0$ then $\tilde{a} = 0$ with certainty, and and if $a = 1$ and N is even, then $\tilde{a} = 1$ with certainty.

Notice that the amplitude θ can hence be recovered via the relation $\theta = \arcsin\sqrt{\theta_a}$ as described above which incurs an ε-error for θ (c.f., Lemma 7, [179]).

A.2   The Hadamard test

Here we present an easy subroutine to evaluate the trace of products of unitary operators U with a density matrix ρ, which is known as the Hadamard test.

Note that this procedure can easily be adapted to be used for ρ being some Gibbs distribution. We then would use a Gibbs state preparation routine in step (2). For example for the evaluation of the gradient of the variational bound, we require this subroutine to evaluate $U = \partial_\theta H$ for ρ being the Gibbs distribution corresponding to the Hamiltonian H.

Appendix B

Appendix 2: Deferred proofs

First for convenience, we formally define quantum Boltzmann machines below.

Definition 7. *A quantum Boltzmann machine to be a quantum mechanical system that acts on a tensor product of Hilbert spaces $\mathcal{H}_v \otimes \mathcal{H}_h \in \mathbb{C}^{2^n}$ that correspond to the visible and hidden subsystems of the Boltzmann machine. It further has a Hamiltonian of the form $H \in \mathbb{C}^{2^n \times 2^n}$ such that $\|H - diag(H)\| > 0$. The quantum Boltzmann machine takes these parameters and then outputs a state of the form $\mathrm{Tr}_h \left(\frac{e^{-H}}{\mathrm{Tr}(e^{-H})} \right)$.*

Given this definition, we are then able to discuss the gradient of the relative entropy between the output of a quantum Boltzmann machines and the input data that it is trained with.

B.1   Derivation of the variational bound

Proof of Lemma 16. Recall that we assume that the Hamiltonian H takes the form

$$H := \sum_k \theta_k v_k \otimes h_k,$$

where v_k and h_k are operators acting on the visible and hidden units respectively and we can assume $h_k = d_k$ to be diagonal in the chosen basis. Under the assumption that $[h_i, h_j] = 0, \forall i, j$, c.f. the assumptions in (5.9), there exists a basis $\{|h\rangle\}$ for the hidden subspace such that $h_k |h\rangle = E_{h,k} |h\rangle$. With these assumptions we can hence

reformulate the logarithm as

$$\log \mathrm{Tr}_h \left[e^{-H} \right] = \log \left(\sum_{v,v',h} \langle v,h| \, e^{-\sum_k \theta_k v_k \otimes h_k} \, |v',h\rangle \, |v\rangle \, \langle v'| \right) \tag{B.1}$$

$$= \log \left(\sum_{v,v',h} \langle v| \, e^{-\sum_k E_{h,k} \theta_k v_k} \, |v'\rangle \, |v\rangle \, \langle v'| \right) \tag{B.2}$$

$$= \log \left(\sum_h e^{-\sum_k E_{h,k} \theta_k v_k} \right), \tag{B.3}$$

where it is important to note that v_k are operators and we hence just used the matrix representation of these in the last step. In order to further simplify this expression, first note that each term in the sum is a positive semi-definite operator. In particularly, note that the matrix logarithm is operator concave and operator monotone, and hence by Jensen's inequality, for any sequence of non-negative number $\{\alpha_i\} : \sum_i \alpha_i = 1$ we have that

$$\log \left(\frac{\sum_{i=1}^N \alpha_i U_i}{\sum_j \alpha_j} \right) \geq \frac{\sum_{i=1}^N \alpha_i \log(U_i)}{\sum_j \alpha_j}.$$

and since we are optimizing $\mathrm{Tr}(\rho \log \rho) - \mathrm{Tr}(\rho \log \sigma_v)$ we hence obtain for arbitrary choice of $\{\alpha_i\}_i$ under the above constraints,

$$\mathrm{Tr} \left(\rho \log \left(\sum_{h=1}^N e^{-\sum_k E_{h,k} \theta_k v_k} \right) \right) = \mathrm{Tr} \left(\rho \log \left(\sum_{h=1}^N \alpha_h \frac{e^{-\sum_k E_{h,k} \theta_k v_k} / \alpha_h}{\sum_{h'} \alpha_{h'}} \right) \right)$$

$$\geq -\mathrm{Tr} \left(\rho \frac{\sum_h \alpha_h \sum_k E_{h,k} \theta_k v_k + \sum_h \alpha_h \log \alpha_h}{\sum_{h'} \alpha_{h'}} \right). \tag{B.4}$$

Hence, the variational bound on the objective function for any $\{\alpha_i\}_i$ is

$$\mathscr{O}_\rho(H) = \mathrm{Tr}(\rho \log \rho) - \mathrm{Tr}(\rho \log \sigma_v)$$

$$\leq \mathrm{Tr}(\rho \log \rho) + \mathrm{Tr} \left(\rho \frac{\sum_h \alpha_h \sum_k E_{h,k} \theta_k v_k + \sum_h \alpha_h \log \alpha_h}{\sum_{h'} \alpha_{h'}} \right) + \log Z =: \tilde{S} \tag{B.5}$$

$\square$

B.2 Gradient estimation

For the following result we will rely on a variational bound in order to train the quantum Boltzmann machine weights for a Hamiltonian H of the form given in (5.10). We begin by proving Lemma 17 in the main work, which will give us an upper bound for the gradient of the relative entropy.

Proof of Lemma 17. We first derive the gradient of the normalization term (Z) in the relative entropy, which can be trivially evaluated using Duhamels formula to obtain

$$\frac{\partial}{\partial \theta_p} \log \mathrm{Tr}\left(e^{-H}\right) = -\mathrm{Tr}\left(\frac{\partial H}{\partial \theta_p}\frac{e^{-H}}{Z}\right) = -\mathrm{Tr}\left(\sigma \partial_{\theta_p} H\right).$$

Note that we can easily evaluate this term by first preparing the Gibbs state $\sigma_{Gibbs} := e^{-H}/Z$ and then evaluating the expectation value of the operator $\partial_{\theta_p} H$ w.r.t. the Gibbs state, using amplitude estimation for the Hadamard test. If T_{Gibbs} is the query complexity for the Gibbs state preparation, the query complexity of the whole algorithm including the phase estimation step is then given by $O(T_{Gibbs}/\tilde{\varepsilon})$ for an $\tilde{\varepsilon}$-accurate estimate of phase estimation. Taking into account the desired accuracy and the error propagation will hence straight forward give the computational complexity to evaluate this part.

We now proceed with the gradient evaluations for the model term. Using the variational bound on the objective function for any $\{\alpha_i\}_i$, given in eq. B.5, we obtain the gradient

$$\frac{\partial \tilde{S}}{\partial \theta_p} = -\mathrm{Tr}\left(\frac{\partial H}{\partial \theta_p}\frac{e^{-H}}{Z}\right) + \mathrm{Tr}\left(\frac{\partial}{\partial \theta_p}\rho \sum_h \alpha_h \sum_k E_{h,k}\theta_k v_k\right) + \frac{\partial}{\partial \theta_p}\sum_h \alpha_h \log \alpha_h$$

$$\tag{B.6}$$

$$= -\mathrm{Tr}\left(\frac{\partial H}{\partial \theta_p}\frac{e^{-H}}{Z}\right) + \frac{\partial}{\partial \theta_p}\left(\sum_h \alpha_h \mathrm{Tr}\left(\rho \sum_k E_{h,k}\theta_k v_k\right) + \sum_h \alpha_h \log \alpha_h\right) \tag{B.7}$$

where the first term results from the partition sum. The latter term can be seen as a new effective Hamiltonian, while the latter term is the entropy. The latter term hence resembles the free energy $F(h) = E(h) - TS(h)$, where $E(h)$ is the mean energy of the effective system with energies $E(h) := \mathrm{Tr}\left(\rho \sum_k E_{h,k}\theta_k v_k\right)$, T the temperature

and $S(h)$ the Shannon entropy of the α_h distribution. We now want to choose these α_h terms to minimize this variational upper bound. It is well-established in statistical physics, see for example [180], that the distribution which maximizes the free energy is the Boltzmann (or Gibbs) distribution, i.e.,

$$\alpha_h = \frac{e^{-\mathrm{Tr}\left(\rho \tilde{H}_h\right)}}{\sum_h e^{-\mathrm{Tr}\left(\rho \tilde{H}_h\right)}},$$

where $\tilde{H}_h := \sum_k E_{h,k}\theta_k v_k$ is a new effective Hamiltonian on the visible units, and the $\{\alpha_i\}$ are given by the corresponding Gibbs distribution for the visible units.

Therefore, our gradients can be taken with respect to this distribution and the bound above, where $\mathrm{Tr}\left(\rho \tilde{H}_h\right)$ is the mean energy of the the effective visible system w.r.t. the data-distribution. For the derivative of the energy term we obtain

$$\frac{\partial}{\partial \theta_p} \sum_h \alpha_h \mathrm{Tr}\left(\rho \sum_k E_{h,k}\theta_k v_k\right) = \tag{B.8}$$

$$= \sum_h \left(\alpha_h \left(\mathbb{E}_{h'}\left[\mathrm{Tr}\left(\rho E_{h',p}v_p\right)\right] - \mathrm{Tr}\left(\rho E_{h,p}v_p\right)\right)\mathrm{Tr}\left(\rho \tilde{H}_h\right) + \alpha_h \mathrm{Tr}\left(\rho E_{h,p}v_p\right)\right) \tag{B.9}$$

$$= \mathbb{E}_h \left[\left(\mathbb{E}_{h'}\left[\mathrm{Tr}\left(\rho E_{h',p}v_p\right)\right] - \mathrm{Tr}\left(\rho E_{h,p}v_p\right)\right)\mathrm{Tr}\left(\rho \tilde{H}_h\right) + \mathrm{Tr}\left(\rho E_{h,p}v_p\right)\right], \tag{B.10}$$

while the entropy term yields

$$\frac{\partial}{\partial \theta_p} \sum_h \alpha_h \log \alpha_h = \sum_h \alpha_h \left(\left[\mathrm{Tr}\left(\rho E_{h,p}v_p\right) - \mathbb{E}_{h'}\left[\mathrm{Tr}\left(\rho E_{h',p}v_p\right)\right]\right]\mathrm{Tr}\left(\rho \tilde{H}_h\right) - \mathrm{Tr}\left(\rho E_{h,p}v_p\right)\right)$$

$$+ \sum_h \alpha_h \left(\mathrm{Tr}\left(\rho E_{h,p}\right) - \mathbb{E}_{h'}\left[\mathrm{Tr}\left(\rho E_{h',p}v_p\right)\right]\right)\log \mathrm{Tr}\left(e^{-\tilde{H}_h}\right)$$

$$+ \mathbb{E}_{h'}\left[\mathrm{Tr}\left(\rho E_{h',p}v_p\right)\right]. \tag{B.11}$$

This can be further simplified to

$$\sum_h \alpha_h \left(\mathrm{Tr}\left(\rho E_{h,p}v_p\right) - \mathbb{E}_{h'}\left[\mathrm{Tr}\left(\rho E_{h',p}v_p\right)\right]\right)\mathrm{Tr}\left(\rho \tilde{H}_h\right) \tag{B.12}$$

$$= \mathbb{E}_h \left[\left(\mathrm{Tr}\left(\rho E_{h,p}v_p\right) - \mathbb{E}_{h'}\left[\mathrm{Tr}\left(\rho E_{h',p}v_p\right)\right]\right)\mathrm{Tr}\left(\rho \tilde{H}_h\right)\right]. \tag{B.13}$$

Te resulting gradient for the variational bound for the visible terms is hence given by

$$\frac{\partial \tilde{S}}{\partial \theta_p} = \mathbb{E}_h \left[\mathrm{Tr} \left(\rho E_{h,p} v_p \right) \right] - \mathrm{Tr} \left(\frac{\partial H}{\partial \theta_p} \frac{e^{-H}}{Z} \right) \tag{B.14}$$

$\square$

Notably, if we consider no interactions between the visible and the hidden layer, then indeed the gradient above reduces recovers the gradient for the visible Boltzmann machine, which was treated in [151], resulting in the gradient

$$\mathrm{Tr} \left(\rho \partial_{\theta_p} H \right) - \mathrm{Tr} \left(\frac{e^{-H}}{Z} \partial_{\theta_p} H \right),$$

under our assumption on the form of H, $\partial_{\theta_p} H = v_p$.

B.2.1 Operationalizing the gradient based training

From Lemma 17, we know that the derivative of the relative entropy w.r.t. any parameter θ_p can be stated as

$$\frac{\partial \tilde{S}}{\partial \theta_p} = \mathbb{E}_h \left[E_{h,p} \right] \mathrm{Tr} \left(\rho v_p \right) - \mathrm{Tr} \left(\frac{\partial H}{\partial \theta_p} \frac{e^{-H}}{Z} \right). \tag{B.15}$$

Since evaluating the latter part is, as mentioned above, straight forward, we give here an algorithm for evaluating the first part.

Now note that we can evaluate each term $\mathrm{Tr} \left(\rho v_k \right)$ individually for all $k \in [D]$, i.e., all D dimensions of the gradient via the Hadamard test for v_k, assuming v_k is unitary. More generally, for non-unitary v_k we could evaluate this term using a linear combination of unitary operations. Therefore, the remaining task is to evaluate the terms $\mathbb{E}_h \left[E_{h,p} \right]$ in (B.15), which reduces to sampling according to the distribution $\{\alpha_h\}$.

For this we need to be able to create a Gibbs distribution for the effective Hamiltonian $\tilde{H}_h = \sum_k \theta_k \mathrm{Tr} \left(\rho v_k \right) h_k$ which contains only D terms and can hence be evaluated efficiently as long as D is small, which we can generally assume to be true. In order to sample according to the distribution $\{\alpha_h\}$, we first evaluate the factors $\theta_k \mathrm{Tr} \left(\rho v_k \right)$

in the sum over k via the Hadamard test, and then use these in order to implement the Gibbs distribution $\exp\left(-\tilde{H}_h\right)/\tilde{Z}$ for the Hamiltonian

$$\tilde{H}_h = \sum_k \theta_k \text{Tr}\left(\rho v_k\right) h_k.$$

In order to do so, we adapt the results of [175] in order to prepare the corresponding Gibbs state (although alternative methods can also be used [181, 182, 183]).

Theorem 15 (Gibbs state preparation [175]). *Suppose that $I \preceq H$ and we are given $K \in \mathbb{R}_+$ such that $\|H\| \leq 2K$, and let $H \in \mathbb{C}^{N \times N}$ be a d-sparse Hamiltonian, and we know a lower bound $z \leq Z = \text{Tr}\left(e^{-H}\right)$. If $\varepsilon \in (0, 1/3)$, then we can prepare a purified Gibbs state $|\gamma\rangle_{AB}$ such that*

$$\left\| \text{Tr}_B\left[|\gamma\rangle\langle\gamma|_{AB}\right] - \frac{e^{-H}}{Z} \right\| \leq \varepsilon \tag{B.16}$$

using

$$\tilde{\mathscr{O}}\left(\sqrt{\frac{N}{z}} K d \log\left(\frac{K}{\varepsilon}\right) \log\left(\frac{1}{\varepsilon}\right) \right) \tag{B.17}$$

queries, and

$$\tilde{\mathscr{O}}\left(\sqrt{\frac{N}{z}} K d \log\left(\frac{K}{\varepsilon}\right) \log\left(\frac{1}{\varepsilon}\right) \left[\log(N) + \log^{5/2}\left(\frac{K}{\varepsilon}\right) \right] \right) \tag{B.18}$$

gates.

Note that by using the above algorithm with $\tilde{H}_{sim}/2$, the preparation of the purified Gibbs state will leave us in the state

$$|\psi\rangle_{Gibbs} := \sum_h \frac{e^{-E_h/2}}{\sqrt{Z}} |h\rangle_A |\phi_h\rangle_B, \tag{B.19}$$

where $|\phi_j\rangle_B$ are mutually orthogonal trash states, which can typically be chosen to be $|h\rangle$, i.e., a copy of the first register, which is irrelevant for our computation, and $|h\rangle_A$ are the eigenstates of $\tilde{H}$. Tracing out the second register will hence leave us in

the corresponding Gibbs state

$$\sigma_h := \sum_h \frac{e^{-E_h}}{Z} |h\rangle \langle h|_A \,,$$

and we can hence now use the Hadamard test with input h_k and σ_h, i.e., the operators on the hidden units and the Gibbs state, and estimate the expectation value $\mathbb{E}_h\left[E_{h,k}\right]$. We provide such a method below.

Proof of Theorem 12. Conceptually, we perform the following steps, starting with Gibbs state preparation followed by a Hadamard test coupled with amplitude estimation to obtain estimates of the probability of a 0 measurement. The proof follows straight from the algorithm described in 5.

From this we see that the runtime constitutes the query complexity of preparing the Gibbs state

$$T_{Gibbs}^V = \tilde{\mathscr{O}}\left(\sqrt{\frac{2^n}{Z}} \frac{\|H(\theta)\| d}{\varepsilon} \log\left(\frac{\|H(\theta)\|}{\tilde{\varepsilon}} \right) \log\left(\frac{1}{\tilde{\varepsilon}} \right) \right),$$

where 2^n is the dimension of the Hamiltonian, as given in Theorem 12 and combining it with the query complexity of the amplitude estimation procedure, i.e., $1/\varepsilon$. However, in order to obtain a final error of ε, we will also need to account for the error in the Gibbs state preparation. For this, note that we estimate terms of the form $\mathrm{Tr}_{AB}\left[\langle\psi|_{Gibbs} (h_k \otimes I) |\psi\rangle_{Gibbs}^V \right] = \mathrm{Tr}_{AB}\left[(h_k \otimes I) |\psi\rangle_{Gibbs}^V \langle\psi|_{Gibbs}^V \right]$. We can hence estimate the error w.r.t. the true Gibbs state σ_{Gibbs} as

$$\begin{aligned}
\mathrm{Tr}_{AB}&\left[(h_k \otimes I) |\psi\rangle_{Gibbs}^V \langle\psi|_{Gibbs}^V \right] - \mathrm{Tr}_A\left[h_k \sigma_{Gibbs} \right] \\
&= \mathrm{Tr}_A\left[h_k \mathrm{Tr}_B\left[|\psi\rangle_{Gibbs}^V \langle\psi|_{Gibbs}^V \right] - h_k \sigma_{Gibbs} \right] \\
&\leq \sum_i \sigma_i(h_k) \left\| \mathrm{Tr}_B\left[|\psi\rangle_{Gibbs}^V \langle\psi|_{Gibbs}^V \right] - \sigma_{Gibbs} \right\| \\
&\leq \tilde{\varepsilon} \sum_i \sigma_i(h_k).
\end{aligned} \tag{B.20}$$

For the final error being less then ε, the precision we use in the phase estimation procedure, we hence need to set $\tilde{\varepsilon} = \varepsilon/(2\sum_i \sigma_i(h_k)) \leq 2^{-n-1}\varepsilon$, reminding that h_k is

unitary, and similarly precision $\varepsilon/2$ for the amplitude estimation, which yields the query complexity of

$$\mathscr{O}\left(\sqrt{\frac{N_h}{z_h}}\frac{\|H(\theta)\|d}{\varepsilon}\left(n^2+n\log\left(\frac{\|H(\theta)\|}{\varepsilon}\right)+n\log\left(\frac{1}{\varepsilon}\right)+\log\left(\frac{\|H(\theta)\|}{\varepsilon}\right)\log\left(\frac{1}{\varepsilon}\right)\right)\right),$$
$$\in\widetilde{O}\left(\sqrt{\frac{N_h}{z_h}}\left(\frac{n^2\|\theta\|_1 d}{\varepsilon}\right)\right). \tag{B.21}$$

where we denote with A the hidden subsystem with dimensionality $2^{n_h}\leq 2^N$, on which we want to prepare the Gibbs state and with B the subsystem for the trash state.

Similarly, for the evaluation of the second part in (B.15) requires the Gibbs state preparation for H, the Hadamard test and phase estimation. Similar as above we meed to take into account the error. Letting the purified version of the Gibbs state for H be given by $|\psi\rangle_{Gibbs}$, which we obtain using Theorem 15, and σ_{Gibbs} be the perfect state, then the error is given by

$$\mathrm{Tr}_{AB}\left[(v_k\otimes h_k\otimes I)\,|\psi\rangle_{Gibbs}\langle\psi|_{Gibbs}\right]-\mathrm{Tr}_A\left[(v_k\otimes h_k)\sigma_{Gibbs}\right]$$
$$=\mathrm{Tr}_A\left[(v_k\otimes h_k)\mathrm{Tr}_B\left[|\psi\rangle^V_{Gibbs}\langle\psi|^V_{Gibbs}\right]-(v_k\otimes h_k)\sigma_{Gibbs}\right]$$
$$\leq\sum_i\sigma_i(v_k\otimes h_k)\left\|\mathrm{Tr}_B\left[|\psi\rangle^V_{Gibbs}\langle\psi|^V_{Gibbs}\right]-h_k\sigma_{Gibbs}\right\|$$
$$\leq\tilde{\varepsilon}\sum_i\sigma_i(v_k\otimes h_k), \tag{B.22}$$

where in this case A is the subsystem of the visible and hidden subspace and B the trash system. We hence upper bound the error similar as above and introducing $\xi:=\max[N/z,N_h/z_h]$ we can find a uniform bound on the query complexity for evaluating a single entry of the D-dimensional gradient is in

$$\widetilde{O}\left(\sqrt{\xi}\left(\frac{n^2\|\theta\|_1 d}{\varepsilon}\right)\right),$$

thus we attain the claimed query complexity by repeating this procedure for each of the D components of the estimated gradient vector $\mathscr{S}$.

Note that we also need to evaluate the terms $\mathrm{Tr}(\rho v_k)$ to precision $\hat{\varepsilon}\leq\varepsilon$, which

though only incurs an additive cost of D/ε to the total query complexity, since this step is required to be performed once. Note that $|\mathbb{E}_h(h_p)| \leq 1$ because h_p is assumed to be unitary. To complete the proof we only need to take the success probability of the amplitude estimation process into account. For completeness we state the algorithm in the appendix and here refer only to Theorem 14, from which we have that the procedure succeeds with probability at least $8/\pi^2$. In order to have a failure probability of the final algorithm of less than $1/3$, we need to repeat the procedure for all d dimensions of the gradient and take the median. We can bound the number of repetitions in the following way.

Let n_f be the number of instances of the gradient estimate such that the error is larger than ε and n_s be the number of instances with an error $\leq \varepsilon$ for one dimension of the gradient, and the result that we take is the median of the estimates, where we take $n = n_s + n_f$ samples. The algorithm gives a wrong answer for each dimension if $n_s \leq \left\lfloor \frac{n}{2} \right\rfloor$, since then the median is a sample such that the error is not bound by ε. Let $p = 8/\pi^2$ be the success probability to draw a positive sample, as is the case of the amplitude estimation procedure. Since each instance of the phase estimation algorithm will independently return an estimate, the total failure probability is given by the union bound, i.e.,

$$\Pr_{fail} \leq D \cdot \Pr\left[n_s \leq \left\lfloor \frac{n}{2} \right\rfloor \right] \leq D \cdot e^{-\frac{n}{2p}\left(p-\frac{1}{2}\right)^2} \leq \frac{1}{3}, \tag{B.23}$$

which follows from the Chernoff inequality for a binomial variable with $p > 1/2$, which is given in our case. Therefore, by taking $n \geq \frac{2p}{(p-1/2)^2} \log(3D) = \frac{16}{(8-\pi^2/2)^2} \log(3D) = O(\log(3D))$, we achieve a total failure probability of at most $1/3$.

This is sufficient to demonstrate the validity of the algorithm if

$$\mathrm{Tr}\left(\rho \tilde{H}_h \right) \tag{B.24}$$

is known exactly. This is difficult to do because the probability distribution α_h is not usually known apriori. As a result, we assume that the distribution will be learned

empirically and to do so we will need to draw samples from the purified Gibbs states used as input. This sampling procedure will incur errors. To take such errors into account assume that we can obtain estimates T_h of B.24 with precision δ_t, i.e.,

$$\left| T_h - \text{Tr}\left(\rho \tilde{H}_h\right) \right| \leq \delta_t. \tag{B.25}$$

Under this assumption we can now bound the distance $|\alpha_h - \tilde{\alpha}_h|$ in the following way. Observe that

$$
\begin{aligned}
|\alpha_h - \tilde{\alpha}_h| &= \left| \frac{e^{-\text{Tr}(\rho \tilde{H}_h)}}{\sum_h e^{-\text{Tr}(\rho \tilde{H}_h)}} - \frac{T_h}{\sum_h T_h} \right| \\
&\leq \left| \frac{e^{-\text{Tr}(\rho \tilde{H}_h)}}{\sum_h e^{-\text{Tr}(\rho \tilde{H}_h)}} - \frac{T_h}{\sum_h e^{-\text{Tr}(\rho \tilde{H}_h)}} \right| + \left| \frac{T_h}{\sum_h e^{-\text{Tr}(\rho \tilde{H}_h)}} - \frac{T_h}{\sum_h T_h} \right|,
\end{aligned} \tag{B.26}
$$

and we hence need to bound the following two quantities in order to bound the error. First, we need a bound on

$$\left| e^{-\text{Tr}(\rho \tilde{H}_h)} - e^{-T_h} \right|. \tag{B.27}$$

For this, let $f(s) := T_h (1-s) + \text{Tr}\left(\rho \tilde{H}_h\right) s$, such that eq. B.27 can be rewritten as

$$
\begin{aligned}
\left| e^{-f(1)} - e^{-f(0)} \right| &= \left| \int_0^1 \frac{d}{ds} e^{-f(s)} ds \right| \\
&= \left| \int_0^1 \dot{f}(s) e^{-f(s)} ds \right| \\
&= \left| \int_0^1 \left(\text{Tr}\left(\rho \tilde{H}_h\right) - T_h \right) e^{-f(s)} ds \right| \\
&\leq \delta e^{-\min_s f(s)} \\
&\leq \delta e^{-\text{Tr}(\rho H_h) + \delta} \tag{B.28}
\end{aligned}
$$

and assuming $\delta \leq \log(2)$, this reduces to

$$\left| e^{-f(1)} - e^{-f(0)} \right| \leq 2\delta e^{-\text{Tr}(\rho \tilde{H}_h)}. \tag{B.29}$$

Second, we need the fact that

$$\left| \sum_h e^{-\text{Tr}(\rho \tilde{H}_h)} - \sum_h T_h \right| \le 2\delta \sum_h e^{-\text{Tr}(\rho \tilde{H}_h)}. \tag{B.30}$$

Using this, eq. B.26 can be upper bound by

$$\frac{2\delta e^{-\text{Tr}(\rho \tilde{H}_h)}}{\sum_h e^{-\text{Tr}(\rho \tilde{H}_h)}} + |T_h| \left| \frac{1}{\sum_h e^{-\text{Tr}(\rho \tilde{H}_h)}} - \frac{1}{(1-2\delta) \sum_h e^{-\text{Tr}(\rho \tilde{H}_h)}} \right|$$
$$\le \frac{2\delta e^{-\text{Tr}(\rho \tilde{H}_h)}}{\sum_h e^{-\text{Tr}(\rho \tilde{H}_h)}} + \frac{4\delta |T_h|}{\sum_h e^{-\text{Tr}(\rho \tilde{H}_h)}}, \tag{B.31}$$

where we used that $\delta \le 1/4$. Note that

$$4\delta |T_h| \le 4\delta \left(e^{-\text{Tr}(\rho \tilde{H}_h)} + 2\delta e^{-\text{Tr}(\rho \tilde{H}_h)} \right)$$
$$= e^{-\text{Tr}(\rho \tilde{H}_h)} \left(4\delta + 8\delta^2 \right)$$
$$\le e^{-\text{Tr}(\rho \tilde{H}_h)} (4\delta + 2\delta)$$
$$\le 6\delta e^{-\text{Tr}(\rho \tilde{H}_h)}, \tag{B.32}$$

which leads to a final error of

$$|\alpha_h - \tilde{\alpha}_h| \le 8\delta \frac{e^{-\text{Tr}(\rho \tilde{H}_h)}}{\sum_h e^{-\text{Tr}(\rho \tilde{H}_h)}}. \tag{B.33}$$

With this we can now bound the error in the expectation w.r.t. the faulty distribution for some function $f(h)$ to be

$$\left| \mathbb{E}_h(f(h)) - \tilde{\mathbb{E}}_h(f(h)) \right| \le 8\delta \sum_h \frac{f(h)e^{-\text{Tr}(\rho \tilde{H}_h)}}{\sum_h e^{-\text{Tr}(\rho \tilde{H}_h)}}$$
$$\le 8\delta \max_h f(h). \tag{B.34}$$

We can hence use this in order to estimate the error introduced in the first term of

eq. B.15 through errors in the distribution $\{\alpha_h\}$ as

$$\left| \mathbb{E}_h[E_{h,p}] \text{Tr}(\rho v_p) - \tilde{\mathbb{E}}[E_{h,p}\text{Tr}(\rho v_p)] \right| \leq 8\delta \max_h |E_{h,p}\text{Tr}(\rho v_p)|$$

$$\leq 8\delta \max_{h,p} |E_{h,p}|, \qquad (B.35)$$

where we used in the last step the unitarity of v_k and the Von-Neumann trace inequality. For an final error of ε, we hence choose $\delta_t = \varepsilon / [16 \max_{h,p} |E_{h,p}|]$ to ensure that this sampling error incurrs at most half the error budget of ε. Thus we ensure $\delta \leq 1/4$ if $\varepsilon \leq 4 \max_{h,p} |E_{h,p}|$.

We can improve the query complexity of estimating the above expectation by values by using amplitude amplification, sice we obtain the measurement via a Hadamard test. For this case we require only $O(\max_{h,p} |E_{h,p}|/\varepsilon)$ samples in order to achieve the desired accuracy from the sampling. Noting that we might not be able to even access $\tilde{H}_h$ without any error, we can deduce that the error of the individual terms of $\tilde{H}_h$ for an ε-error in the final estimate must be bounded by $\delta_t v \|\theta\|_1$, where with abuse of notation, δ_t now denotes the error in the estimates of $E_{h,k}$. Even taking this into account, the evaluation of this contribution is however dominated by the second term, and hence can be neglected in the analysis.

□

B.3 Approach 2: Divided Differences

In this section we develop a scheme to train a quantum Boltzmann machine using divided difference estimates for the relative entropy error. The idea for this is straightforward: First we construct an interpolating polynomial from the data. Second, an approximation of the derivative at tany point can be then obtained by a direct differentiation of the interpolant. We assume in the following that we can simulate and evaluate $\text{Tr}(\rho \log \sigma_v)$. As this is generally non-trivial, and the error is typically large, we propose in the next section a different more specialised approach which, however, still allows us to train arbitrary models with the relative entropy objective.

In order to proof the error of the gradient estimation via interpolation, we first

need to establish error bounds on the interpolating polynomial which can be obtained via the remainder of the Lagrange interpolation polynomial. The gradient error for our objective can then be obtained by as a combination of this error with a bound on the $n+1$-st order derivative of the objective. We start by bounding the error in the polynomial approximation.

Lemma 18. *Let $f(\theta)$ be the $n+1$ times differentiable function for which we want to approximate the gradient and let $p_n(\theta)$ be the degree n Lagrange interpolation polynomial for points $\{\theta_1, \theta_2, \ldots, \theta_k, \ldots, \theta_n\}$. The gradient evaluated at point θ_k is then given by the interpolation polynomial*

$$\frac{\partial p(\theta_k)}{\partial \theta} = \sum_{j=0}^{n} f(\theta_j) \mathscr{L}'_{n,j}(\theta_k), \tag{B.36}$$

where $\mathscr{L}'_{n,j}$ is the derivative of the Lagrange interpolation polynomials $\mathscr{L}_{\mu,j}(\theta) := \prod_{\substack{k=0 \\ k \neq j}}^{\mu} \frac{\theta - \theta_k}{\theta_j - \theta_k}$, and the error is given by

$$\left| \frac{\partial f(\theta_k)}{\partial \theta} - \frac{\partial p_n(\theta_k)}{\partial \theta} \right| \leq \frac{1}{(n+1)!} \left| f^{(n+1)}(\xi(\theta_k)) \prod_{\substack{j=0 \\ j \neq k}}^{n} (\theta_j - \theta_k) \right|, \tag{B.37}$$

where $\xi(\theta_k)$ is a constant depending on the point θ_k at which we evaluated the gradient, and $f^{(i)}$ denotes the i-th derivative of f.

Note that θ is a point within the set of points at which we evaluate.

Proof. Recall that the error for the degree n Lagrange interpolation polynomial is given by

$$f(\theta) - p_n(\theta) \leq \frac{1}{(n+1)!} f^{(n+1)}(\xi_\theta) w(\theta), \tag{B.38}$$

where $w(\theta) := \prod_{j=1}^{n} (\theta - \theta_j)$. We want to estimate the gradient of this, and hence need to evaluate

$$\frac{\partial f(\theta)}{\partial \theta} - \frac{\partial p_n(\theta)}{\partial \theta} \leq \lim_{\Delta \to 0} \left(\frac{\frac{1}{(n+1)!} f^{(n+1)}(\xi_{\theta+\Delta}) w(\theta+\Delta) - \frac{1}{(n+1)!} f^{(n+1)}(\xi_\theta) w(\theta)}{\Delta} \right). \tag{B.39}$$

Now, since we do not necessarily want to estimate the gradient at an arbitrary point θ but indeed have the freedom to choose the point, we can set θ to be one of the points at which we evaluate the function $f(\theta)$, i.e., $\theta \in \{\theta_i\}_{i=1}^{n}$. Let this choice be given by θ_k, arbitrarily chosen. Then we see that the latter term vanishes since $w(\theta_k) = 0$. Therefore we have

$$\frac{\partial f(\theta_k)}{\partial \theta} - \frac{\partial p_n(\theta_k)}{\partial \theta} \leq \lim_{\Delta \to 0} \left(\frac{\frac{1}{(n+1)!} f^{(n+1)}(\xi_{\theta_k+\Delta}) w(\theta_k+\Delta)}{\Delta} \right), \tag{B.40}$$

and noting that $w(\theta_k)$ contains one term $(\theta_k + \Delta - \theta_k) = \Delta$ achieves the claimed result. $\square$

We will perform a number of approximation steps in order to obtain a form which can be simulated on a quantum computer more efficiently, and only then resolve to divided differences at this "lower level". In detail we will perform the following steps. As described in the body of the thesis, we perform the following steps in order to obtain the gradient.

1. Approximate the logarithm via a Fourier-like approximation

$$\log \sigma_v \to \log_{K,M} \sigma_v, \tag{B.41}$$

which yields a Fourier-like series $\sum_m c_m \exp(im\pi\sigma_v)$.

2. Evaluate the gradient of $\mathrm{Tr}\left(\frac{\partial}{\partial \theta} \rho \log_{K,M}(\sigma_v) \right)$, yielding terms of the form

$$\int_0^1 ds e^{(ism\pi\sigma_v)} \frac{\partial \sigma_v}{\partial \theta} e^{(i(1-s)m\pi\sigma_v)}. \tag{B.42}$$

3. Each term in this expansion can be evaluated separately via a sampling procedure, i.e.,

$$\int_0^1 ds e^{(ism\pi\sigma_v)} \frac{\partial \sigma_v}{\partial \theta} e^{(i(1-s)m\pi\sigma_v)} \approx \mathbb{E}_s \left[e^{(ism\pi\sigma_v)} \frac{\partial \sigma_v}{\partial \theta} e^{(i(1-s)m\pi\sigma_v)} \right]. \tag{B.43}$$

4. Apply a divided difference scheme to approximate the gradient $\frac{\partial \sigma_v}{\partial \theta}$.

5. Use the Fourier series approach to aproximate the density operator σ_v by the series of itself, i.e., $\sigma_v \approx F(\sigma_v) := \sum_{m'} c_{m'} \exp(im\pi m'\sigma_v)$.

6. Evaluate these terms conveniently via sample based Hamiltonian simulation and the Hadamard test.

In the following we will give concrete bounds on the error introduced by the approximations and details of the implementation. The final result is then stated in Theorem 13. We first bound the error in the approximation of the logarithm and then use Lemma 37 of [175] to obtain a Fouries series approximation which is close to $\log(z)$. The Taylor series of

$$\log(x) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{(x-1)^k}{k} = \sum_{k=1}^{K_1} (-1)^{k+1} \frac{(x-1)^k}{k} + R_{K_1+1}(x-1),$$

for $x \in (0,1)$ and where $R_{K+1}(z) = \frac{f^{K_1+1}(c)}{K!}(z-c)^{K_1}z$ is the Cauchy remainder of the Taylor series, for $-1 < z < 0$. The error can hence be bounded as

$$|R_{K_1+1}(z)| = \left| (-1)^{K_1} \frac{z^{K_1+1}(1-\alpha)^{K_1}}{(1+\alpha z)^{K_1+1}} \right|,$$

where we evaluated the derivatives of the logarithm and $0 \leq \alpha \leq 1$ is a parameter. Using that $1 + \alpha z \geq 1 + z$ (since $z \leq 0$) and hence $0 \leq \frac{1-\alpha}{1+\alpha z} \leq 1$, we obtain the error bound

$$|R_{K_1+1}(z)| \leq \frac{|z|^{K_1+1}}{1+z} \tag{B.44}$$

Reversing to the variable x the error bound for the Taylor series, and assuming that $0 < \delta_l < z$ and $0 < |1-z| \leq \delta_u < 1$, which is justified if we are dealing with sufficiently mixed states, then the approximation error is given by

$$|R_{K_1+1}(z)| \leq \frac{(\delta_l)^{K_1+1}}{\delta_u} \stackrel{!}{\leq} \varepsilon_1. \tag{B.45}$$

Hence in order to achieve the desired error ε_1 we need

$$K_1 \geq \frac{\log\left((\varepsilon_1 \delta_u)^{-1}\right)}{\log\left((\delta_l)^{-1}\right)}.$$

We hence can chose K_1 such that the error in the approximation of the Taylor series is $\leq \varepsilon_1/4$. This implies we can make use of Lemma 37 of [175], and therefore obtain a Fourier series approximation for the logarithm. We will restate this Lemma here for completeness:

Lemma 19 (Lemma 37, [175]). *Let* $f : \mathbb{R} \to \mathbb{C}$ *and* $\delta, \varepsilon \in (0,1)$, *and* $T(f) := \sum_{k=0}^{K} a_k x^k$ *be a polynomial such that* $|f(x) - T(f)| \leq \varepsilon/4$ *for all* $x \in [-1+\delta, 1-\delta]$. *Then* $\exists c \in \mathbb{C}^{2M+1}$:

$$\left| f(x) - \sum_{m=-M}^{M} c_m e^{\frac{i\pi m}{2}x} \right| \leq \varepsilon \tag{B.46}$$

for all $x \in [-1+\delta, 1-\delta]$, *where* $M = \max\left(2\left\lceil \ln\left(\frac{4\|a\|_1}{\varepsilon}\right)\frac{1}{\delta}\right\rceil, 0\right)$ *and* $\|c\|_1 \leq \|a\|_1$. *Moreover, c can be efficiently calculated on a classical computer in time* $\text{poly}(K, M, \log(1/\varepsilon))$.

In order to apply this lemma to our case, we restrict the approximation rate to the range (δ_l, δ_u), where $0 < \delta_l \leq \delta_u < 1$. Therefore we obtain over this range a approximation of the following form.

Corollary 16. *Let* $f : \mathbb{R} \to \mathbb{C}$ *be defined as* $f(x) = \log(x)$, $\delta, \varepsilon_1 \in (0,1)$, *and* $\log_K(1-x) := \sum_{k=1}^{K_1} \frac{(-1)^{k-1}}{k}x^k$ *such that* $a_k := \frac{(-1)^{k-1}}{k}$ *and* $\|a\|_1 = \sum_{k=1}^{K_1}\frac{1}{k}$ *with* $K_1 \geq \frac{\log\left(4(\varepsilon_1\delta_1^u)^{-1}\right)}{\log\left((\delta_l)^{-1}\right)}$ *such that* $|\log(x) - \log_K(x)| \leq \varepsilon_1/4$ *for all* $x \in [\delta_l, \delta_u]$. *Then* $\exists c \in \mathbb{C}^{2M+1}$:

$$\left| f(x) - \sum_{m=-M_1}^{M_1} c_m e^{\frac{i\pi m}{2}x} \right| \leq \varepsilon_1 \tag{B.47}$$

for all $x \in [\delta_l, \delta_u]$, *where* $M_1 = \max\left(2\left\lceil \ln\left(\frac{4\|a\|_1}{\varepsilon_1}\right)\frac{1}{1-\delta_u}\right\rceil, 0\right)$ *and* $\|c\|_1 \leq \|a\|_1$. *Moreover, c can be efficiently calculated on a classical computer in time* $\text{poly}(K_1, M_1, \log(1/\varepsilon_1))$.

Proof. The proof follows straight forward by combining Lemma 19 with the approximation of the logarithm and the range over which we want to approximate the function. $\qquad\square$

In the following we denote with $\log_{K,M}(x) := \sum_{m=-M_1}^{M_1} c_m e^{\frac{i\pi m}{2}x}$, where we keep the K-subscript to denote that classical computation of this approximation

is poly(K)-dependent. We can now express the gradient of the objective via this approximation as

$$\text{Tr}\left(\frac{\partial}{\partial\theta}\rho\log_{K,M}\sigma_v\right) \approx \sum_{m=-M_1}^{M_1}\frac{ic_m m\pi}{2}\int_0^1 ds\,\text{Tr}\left(\rho e^{\frac{is\pi m}{2}\sigma_v}\frac{\partial\sigma_v}{\partial\theta}e^{\frac{i(1-s)\pi m}{2}\sigma_v}\right).$$
(B.48)

where we can evaluate each term in the sum individually and then classically post process the results, i.e., sum these up. In particular the latter can be evaluated as the expectation value over s, i.e.,

$$\int_0^1 ds\,\text{Tr}\left(\rho e^{\frac{is\pi m}{2}\sigma_v}\frac{\partial\sigma_v}{\partial\theta}e^{\frac{i(1-s)\pi m}{2}\sigma_v}\right) = \mathbb{E}_{s\in[0,1]}\left[\text{Tr}\left(\rho e^{\frac{is\pi m}{2}\sigma_v}\frac{\partial\sigma_v}{\partial\theta}e^{\frac{i(1-s)\pi m}{2}\sigma_v}\right)\right],$$
(B.49)

which we can evaluate separately on a quantum device. In the following we hence need to device a method to evaluate this expectation value.

First, we will expand the gradient using a divided difference formula such that $\frac{\partial\sigma_v}{\partial\theta}$ is approximated by the Lagrange interpolation polynomial of degree $\mu-1$, i.e.,

$$\frac{\partial\sigma_v}{\partial\theta}(\theta) \approx \sum_{j=0}^{\mu}\sigma_v(\theta_j)\mathscr{L}'_{\mu,j}(\theta),$$

where

$$\mathscr{L}_{\mu,j}(\theta) := \prod_{\substack{k=0\\k\neq j}}^{\mu}\frac{\theta-\theta_k}{\theta_j-\theta_k}.$$

Note that the order μ is free to chose, and will guarantee a different error in the solution of the gradient estimate as described prior in Lemma 18. Using this in the gradient estimation, we obtain a polynomial of the form (evaluated at θ_j, i.e., the chosen points)

$$\sum_{m=-M_1}^{M_1}\frac{ic_m m\pi}{2}\sum_{j=0}^{\mu}\mathscr{L}'_{\mu,j}(\theta_j)\mathbb{E}_{s\in[0,1]}\left[\text{Tr}\left(\rho e^{\frac{is\pi m}{2}\sigma_v}\sigma_v(\theta_j)e^{\frac{i(1-s)\pi m}{2}\sigma_v}\right)\right],\quad\text{(B.50)}$$

where each term again can be evaluated separately, and efficiently combined via classical post processing. Note that the error in the Lagrange interpolation polynomial decreases exponentially fast, and therefore the number of terms we use is

sufficiently small to do so. Next, we need to deploy a method to evaluate the above expressions. In order to do so, we implement σ_v as a Fourier series of itself, i.e., $\sigma_v = \arcsin(\sin(\sigma_v \pi/2)/(\pi/2))$, which we will then approximate similar to the approach taken in Lemma 19. With this we obtain the following result.

Lemma 20. *Let* $\delta, \varepsilon_2 \in (0,1)$, *and* $\tilde{x} := \sum_{m'=-M_2}^{M_2} \tilde{c}_{m'} e^{i\pi m' x/2}$ *with* $K_2 \geq \frac{\log(4/\varepsilon_2)}{\log(\delta_u^{-1})}$ *and* $M_2 \geq \left\lceil \log\left(\frac{4}{\varepsilon_2}\right) \sqrt{(2\log \delta_u^{-1})^{-1}} \right\rceil$ *and* $x \in [\delta_l, \delta_u]$. *Then* $\exists \tilde{c} \in \mathbb{C}^{2M+1}$:

$$|x - \tilde{x}| \leq \varepsilon_2 \tag{B.51}$$

for all $x \in [\delta_l, \delta_u]$, *and* $\|c\|_1 \leq 1$. *Moreover,* $\tilde{c}$ *can be efficiently calculated on a classical computer in time* $\mathrm{poly}(K_2, M_2, \log(1/\varepsilon_2))$.

Proof. Invoking the technique used in [175], we expand

$$\arcsin(z) = \sum_{k'=0}^{K_2} 2^{-2k'} \binom{2k'}{k'} \frac{z^{2k'+1}}{2k'+1} + R_{K_2+1}(z),$$

wher R_{K_2+1} is the remainder as before. For $0 < z \leq \delta_u \leq 1/2$, remainder can be bound by $|R_{K_2+1}| \leq \frac{|\delta_u|^{K_2+1}}{1/2} \overset{!}{\leq} \varepsilon_2/2$, which gives the bound

$$K_2 \geq \frac{\log(4/\varepsilon_2)}{\log(\delta_u^{-1})}.$$

We then approximate

$$\sin^l(x) = \left(\frac{i}{2}\right)^l \sum_{m'=0}^{l} (-1)^{m'} \binom{l}{m'} e^{ix(2m'-l)} \tag{B.52}$$

by

$$\sin^l(x) \approx \left(\frac{i}{2}\right)^l \sum_{m'=\lceil l/2\rceil - M_2}^{\lfloor l/2\rfloor + M_2} (-1)^{m'} \binom{l}{m'} e^{ix(2m'-l)}, \tag{B.53}$$

which induces an error of $\varepsilon_2/2$ for the choice

$$M_2 \geq \left\lceil \log\left(\frac{4}{\varepsilon_2}\right) \sqrt{(2\log \delta_u^{-1})^{-1}} \right\rceil.$$

This can be seen by using Chernoff's inequality for sums of binomial coefficients, i.e.,

$$\sum_{m'=\lceil l/2+M_2\rceil}^{l} 2^{-l}\binom{l}{m'} \le e^{-\frac{2M_2^2}{l}},$$

and chosing M appropriately. Finally, defining $f(z) := \arcsin(\sin(z\pi/2)/(\pi/2))$, as well as $\tilde{f}_1 := \sum_{k'=0}^{K_2} b_{k'}\sin^{2k'+1}(z\pi/2)$ and

$$\tilde{f}_2(z) := \sum_{k'=0}^{K_2} b_{k'}\left(\frac{i}{2}\right)^l \sum_{m'=\lceil l/2\rceil-M_2}^{\lfloor l/2\rfloor+M_2} (-1)^{m'}\binom{l}{m'} e^{ix(2m'-l)}, \qquad (B.54)$$

and observing that

$$\left\| f-\tilde{f}_2 \right\|_\infty \le \left\| f-\tilde{f}_1 \right\|_\infty + \left\| \tilde{f}_1-\tilde{f}_2 \right\|_\infty,$$

yields the final error of ε_2 for the approximation $z \approx \tilde{z} = \sum_{m'} \tilde{c}_{m'} e^{i\pi m'z/2}$. $\qquad\square$

Note that this immediately leads to an ε_2 error in the spectral norm for the approximation

$$\left\| \sigma_v - \sum_{m'=-M_2}^{M_2} \tilde{c}_{m'} e^{i\pi m'\sigma_v/2} \right\|_2 \le \varepsilon_2, \qquad (B.55)$$

where σ_v is the reduced density matrix.

Since our final goal is to estimate $\mathrm{Tr}\left(\partial_\theta \rho \log \sigma_v\right)$, with a variety of $\sigma_v(\theta_j)$ using the divided difference approach, we also need to bound the error in this estimate which we introduce with the above approximations. Bounding the derivative with respect to the remainder can be done by using the truncated series expansion and bounding the gradient of the remainder. This yields the following result.

Lemma 21. *For the of the parameters $M_1, M_2, K_1, L, \mu, \Delta, s$ given in eq. (B.89-B.96), and ρ, σ_v being two density matrices, we can estimate the gradient of the relative entropy such that*

$$\left| \partial_\theta \mathrm{Tr}\left(\rho \log \sigma_v\right) - \partial_\theta \mathrm{Tr}\left(\rho \log_{K_1, M_1} \tilde{\sigma}_v\right) \right| \le \varepsilon, \qquad (B.56)$$

where the function $\partial_\theta \text{Tr}\left(\rho \log_{K_1,M_1} \tilde{\sigma}_v\right)$ evaluated at θ is defined as

$$\text{Re}\left[\sum_{m=-M_1}^{M_1}\sum_{m'=-M_2}^{M_2}\frac{ic_m\tilde{c}_{m'}m\pi}{2}\sum_{j=0}^{\mu}\mathscr{L}'_{\mu,j}(\theta)\mathbb{E}_{s\in[0,1]}\left[\text{Tr}\left(\rho e^{\frac{is\pi m}{2}\sigma_v}e^{\frac{i\pi m'}{2}\sigma_v(\theta_j)}e^{\frac{i(1-s)\pi m}{2}\sigma_v}\right)\right]\right]$$

(B.57)

The gradient can hence be approximated to error ε with $O(poly(M_1,M_2,K_1,L,s,\Delta,\mu))$ computation on a classical computer and using only the Hadamard test, Gibbs state preparation and LCU on a quantum device.

Notably the expression in (B.57) can now be evaluated with a quantum-classical hybrid device by evaluating each term in the trace separately via a Hadamard test and, since the number of terms is only polynomial, and then evaluating the whole sum efficiently on a classical device.

Proof. For the proof we perform the following steps. Let $\sigma_i(\rho)$ be the singular values of ρ, which are equivalently the eigenvalues since ρ is Hermitian. Then observe that the gradient can be separated in different terms, i.e., let $\log_{K_1,M_1}^s \sigma_v$ be the approximation as given in (B.57) for a finite sample of the expectation values $\mathbb{E}_s$, then we have

$$\left|\partial_\theta \text{Tr}\left(\rho \log \sigma_v\right) - \partial_\theta \text{Tr}\left(\rho \log_{K_1,M_1}^s \tilde{\sigma}_v\right)\right| \leq$$
$$\leq \sum_i \sigma_i(\rho)\cdot\left\|\partial_\theta[\log\sigma_v - \log_{K_1,M_1}^s \tilde{\sigma}_v]\right\|$$
$$\leq \sum_i \sigma_i(\rho)\cdot\left(\left\|\partial_\theta[\log\sigma_v - \log_{K_1,M_1}\sigma_v]\right\|\right.$$
$$\left. + \left\|\partial_\theta[\log_{K_1,M_1}\sigma_v - \log_{K_1,M_1}\tilde{\sigma}_v]\right\| + \left\|\partial_\theta[\log_{K_1,M_1}\tilde{\sigma}_v - \log_{K_1,M_1}^s\tilde{\sigma}_v]\right\|\right) \quad (B.58)$$

where the second step follows from the Von-Neumann trace inequality and the terms are (1) the error in approximating the logarithm, (2) the error introduced by the divided difference and the approximation of σ_v as a Fourier-like series, and (3) is the finite sampling approximation error. We can now bound the different term separately, and start with the first part which is in general harder to estimate. We partition the bound in three terms, corresponding to the three different approximations taken

above.

$$\left\| \partial_\theta [\log \sigma_v - \log_{K_1, M_1} \sigma_v] \right\| \leq$$

$$\leq \left\| \partial_\theta \sum_{k=K_1+1}^{\infty} \frac{(-1)^k}{k} \sigma_v^k \right\| + \left\| \partial_\theta \sum_{k=1}^{K_1} \frac{(-1)^k}{k} \sum_{l=L}^{\infty} b_l^{(k)} \sin^l(\sigma_v \pi/2) \right\|$$

$$+ \left\| \partial_\theta \sum_{k=1}^{K_1} \frac{(-1)^k}{k} \sum_{l=L}^{\infty} b_l^{(k)} \left(\frac{i}{2} \right)^l \sum_{m \in [0, \lceil l/2 \rceil - M_1] \cup [\lfloor l/2 \rfloor + M_1, l]} (-1)^m e^{i(2m-l)\sigma_v \pi/2} \right\|$$

The first term can be bound in the following way:

$$\leq \sum_{k=K_1+1}^{\infty} \|\sigma_v\|^{k-1} = \frac{\|\sigma_v\|^{K_1}}{1 - \|\sigma_v\|}, \tag{B.59}$$

and, assuming $\|\sigma_v\| < 1$, we hence can set

$$K_1 \geq \log((1 - \|\sigma_v\|)\varepsilon/9)/\log(\|\sigma_v\|) \tag{B.60}$$

appropriately in order to achieve an $\varepsilon/9$ error. The second term can be bound by assuming that $\|\sigma_v \pi\| < 1$, and chosing

$$L \geq \log \frac{\left(\frac{\varepsilon}{9\pi K \left\| \frac{\partial \sigma_v}{\partial \theta} \right\|} \right)}{\log(\|\sigma_v\| \pi)},$$

which we derive by observing that

$$\leq \sum_{k=1}^{K} \frac{1}{k} \sum_{l=L}^{\infty} b_l^{(k)} l \left\| \sin^{l-1}(\sigma_v \pi/2) \right\| \cdot \left\| \frac{\pi}{2} \frac{\partial \sigma_v}{\partial \theta} \right\| \tag{B.61}$$

$$< \sum_{k=1}^{K} \frac{1}{k} \sum_{l=L+1}^{\infty} b_l^{(k)} \pi \|\sigma_v \pi\|^{l-1} \cdot \left\| \frac{\partial \sigma_v}{\partial \theta} \right\| \tag{B.62}$$

$$\leq \sum_{k=1}^{K} \frac{1}{k} \pi \|\sigma_v \pi\|^{L} \cdot \left\| \frac{\partial \sigma_v}{\partial \theta} \right\|, \tag{B.63}$$

where we used in the second step that $l < 2^l$. Finally, the last term can be bound

similarly, which yields

$$\leq \sum_{k=1}^{K} \frac{1}{k} \sum_{l=1}^{L} b_l^{(k)} e^{-2(M_1)^2/l} \cdot l \cdot \frac{\pi}{2} \left\| \frac{\partial \sigma_v}{\partial \theta} \right\| \tag{B.64}$$

$$\leq \sum_{k=1}^{K} \frac{L}{k} \sum_{l=1}^{L} b_l^{(k)} e^{-2(M_1)^2/L} \frac{\pi}{2} \left\| \frac{\partial \sigma_v}{\partial \theta} \right\| \tag{B.65}$$

$$\leq \sum_{k=1}^{K} \frac{L}{k} e^{-2(M_1)^2/L} \frac{\pi}{2} \left\| \frac{\partial \sigma_v}{\partial \theta} \right\| \leq \frac{KL\pi}{2} e^{-2(M_1)^2/L} \left\| \frac{\partial \sigma_v}{\partial \theta} \right\|, \tag{B.66}$$

and we can hence chose

$$M_1 \geq \sqrt{L \log \left(\frac{9 \left\| \frac{\partial \sigma_v}{\partial \theta} \right\| K_1 L \pi}{2\varepsilon} \right)}$$

in order to decrease the error to $\varepsilon/3$ for the first term in (B.58).

For the second term, first note that with the notation we chose, $\left\| \partial_\theta [\log_{K_1,M_1} \sigma_v - \log_{K_1,M_1} \tilde{\sigma}_v] \right\|$ is the difference between the log-approximation where the gradient of σ_v is still exact, i.e., (B.48), and the version where we approximate the gradient via divided differences and the linear combination of unitaries, given in (B.57). Recall that the first level approximation was given by

$$\sum_{m=-M_1}^{M_1} \frac{ic_m m\pi}{2} \int_0^1 ds \, \mathrm{Tr} \left(\rho e^{\frac{is\pi m}{2} \sigma_v} \frac{\partial \sigma_v}{\partial \theta} e^{\frac{i(1-s)\pi m}{2} \sigma_v} \right),$$

where we went from the expectation value formulation back to the integral formulation to avoid consideration of potential errors due to sampling.

Bounding the difference hence yields one term from the divided difference approximation of the gradient and an error from the Fourier series, which we can both bound separately. Denoting $\partial \tilde{p}(\theta_k)/\partial \theta$ as the divided difference and the LCU approximation of the Fourier series[1], and with $\partial p(\theta_k)/\partial \theta$ the divided difference

---

[1]which effectively means that we approximate the coefficients of the interpolation polynomial

without approximation via the Fourier series, we hence have

$$\left\| \partial_\theta [\log_{K_1,M_1} \sigma_v - \log_{K_1,M_1} \tilde{\sigma}_v] \right\| \leq \tag{B.67}$$

$$\leq \left| \sum_{m=-M_1}^{M_1} \frac{ic_m m\pi}{2} \int_0^1 ds \, \mathrm{Tr}\left(\rho e^{\frac{is\pi m}{2}\sigma_v} \left(\frac{\partial \sigma_v}{\partial \theta} - \frac{\partial \tilde{p}(\theta_k)}{\partial \theta} \right) e^{\frac{i(1-s)\pi m}{2}\sigma_v} \right) \right| \tag{B.68}$$

$$\leq \frac{M_1 \pi \|a\|_1}{2} \int_0^1 ds \sum_i \sigma_i(\rho) \left\| \frac{\partial \sigma_v}{\partial \theta} - \frac{\partial \tilde{p}(\theta_k)}{\partial \theta} \right\| \tag{B.69}$$

$$\leq \frac{M_1 \pi \|a\|_1}{2} \int_0^1 ds \sum_i \sigma_i(\rho) \left(\left\| \frac{\partial \sigma_v}{\partial \theta} - \frac{\partial p(\theta_k)}{\partial \theta} \right\| + \left\| \frac{\partial p(\theta_k)}{\partial \theta} - \frac{\partial \tilde{p}(\theta_k)}{\partial \theta} \right\| \right) \tag{B.70}$$

$$\leq \frac{M_1 \|a\|_1 \pi}{2} \int_0^1 ds \sum_i \sigma_i(\rho) \left(\left\| \frac{\partial^{\mu+1}\sigma_v}{\partial \theta^{\mu+1}} \right\| \left(\frac{\Delta}{\mu-1} \right)^\mu \frac{\max_k(\mu-k)!}{(\mu+1)!} + \sum_{j=0}^\mu |\mathscr{L}'_{\mu,j}(\theta_j)| \|\sigma_v - \tilde{\sigma}_v\| \right) \tag{B.71}$$

$$\leq \frac{M_1 \|a\|_1 \pi}{2} \left(\left\| \frac{\partial^{\mu+1}\sigma_v}{\partial \theta^{\mu+1}} \right\| \left(\frac{\Delta}{\mu-1} \right)^\mu \frac{\mu!}{(\mu+1)!} + \mu \left\| \mathscr{L}'_{\mu,j}(\theta_j) \right\|_\infty \varepsilon_2 \right), \tag{B.72}$$

where $\|a\|_1 = \sum_{k=1}^{K_1} 1/k$, and we used in the last step the results of Lemma 20. Under appropriate assumptions on the grid-spacing for the divided difference scheme Δ and the number of evaluated points μ as well as a bound on the $\mu + 1$-st derivative of σ_v w.r.t. θ, we can hence also bound this error. In order to do so, we need to analyze the $\mu + 1$-st derivative of $\sigma_v = \mathrm{Tr}_h \left[e^{-H} \right] / Z$ with $Z = \mathrm{Tr}\left(e^{-H} \right)$. For this we have

$$\left\| \frac{\partial^{\mu+1}\sigma_v}{\partial \theta^{\mu+1}} \right\| \leq \sum_{p=1}^{\mu+1} \binom{\mu+1}{p} \left\| \frac{\partial^p \mathrm{Tr}_h \left[e^{-H} \right]}{\partial \theta^p} \right\| \left\| \frac{\partial^{\mu+1-p} Z^{-1}}{\partial \theta^{\mu+1-p}} \right\|$$

$$\leq 2^{\mu+1} \max_p \left\| \frac{\partial^p \mathrm{Tr}_h \left[e^{-H} \right]}{\partial \theta^p} \right\| \left\| \frac{\partial^{\mu+1-p} Z^{-1}}{\partial \theta^{\mu+1-p}} \right\| \tag{B.73}$$

We have that

$$\left\| \frac{\partial^p \mathrm{Tr}_h \left[e^{-H} \right]}{\partial \theta^p} \right\| \leq \dim(H_h) \left\| \frac{\partial^q e^{-H}}{\partial \theta^q} \right\| \tag{B.74}$$

where $\dim(H_h) = 2^{n_h}$. In order to bound this, we take advantage of the infinitesimal

expansion of the exponent, i.e.,

$$
\begin{aligned}
\left\| \frac{\partial^q e^{-H}}{\partial \theta^q} \right\| &= \left\| \frac{\partial^q}{\partial \theta^q} \lim_{r \to \infty} \prod_{j=1}^{r} e^{-H/r} \right\| \\
&= \left\| \lim_{r \to \infty} \left(\frac{\partial^q e^{-H/r}}{\partial \theta^q} \prod_{j=2}^{r} e^{-H/r} + \frac{\partial^{q-1} e^{-H/r}}{\partial \theta^{q-1}} \frac{\partial e^{-H/r}}{\partial \theta} \prod_{j=3}^{r} e^{-H/r} + \cdots \right) \right\| \\
&\leq \lim_{r \to \infty} \left(\left\| \frac{\partial H/r}{\partial \theta} \right\|^q \cdot r^q + O\left(\frac{1}{r}\right) \right) \left\| e^{-H} \right\| = \left\| \frac{\partial H}{\partial \theta} \right\|^q \left\| e^{-H} \right\|, \quad \text{(B.75)}
\end{aligned}
$$

where the last step follows from the fact that we have r^q terms and that we used that the error introduced by the commutations above will be of $O(1/r)$. Observing that $\partial_{\theta_i} H = \partial_{\theta_i} \sum_j \theta_j H_j = H_i$ and assuming that λ_{max} is the largest singular eigenvalue of H, we can hence bound this by $\lambda_{max}^q \left\| e^{-H} \right\|$.

$$
\begin{aligned}
\left\| \frac{\partial^p \text{Tr}_h \left[e^{-H} \right]}{\partial \theta^p} \right\| &\leq \dim(H_h) \left\| \frac{\partial^q e^{-H}}{\partial \theta^q} \right\| \\
&\leq \lambda_{max}^p \dim(H_h) \left\| \text{Tr}_h \left[e^{-H} \right] \right\|, \quad \text{(B.76)}
\end{aligned}
$$

$$
\begin{aligned}
\left\| \frac{\partial^{\mu+1-p} Z^{-1}}{\partial \theta^{\mu+1-p}} \right\| &\leq \frac{(\mu+1-p)! |\lambda_{max}|^{\mu+1-p}}{Z^{\mu+2-p}} \text{Tr}\left(e^{-H} \right) \\
&\leq \left(\frac{\mu+1-p}{eZ} \right)^{\mu+1-p} \frac{e}{Z} |\lambda_{max}|^{\mu+1-p} \text{Tr}\left(e^{-H} \right) \\
&= \left(\frac{\mu+1-p}{eZ} \right)^{\mu+1-p} e |\lambda_{max}|^{\mu+1-p} \quad \text{(B.77)}
\end{aligned}
$$

We can therefore find a bound for (B.73) as

$$
\left\| \frac{\partial^{\mu+1} \sigma_v}{\partial \theta^{\mu+1}} \right\| \leq e 2^{\mu+1+n_h} \lambda_{max}^{\mu+1} \left\| \text{Tr}_h \left[e^{-H} \right] \right\| \max_p \left(\frac{\mu+1-p}{eZ} \right)^{\mu+1-p}. \quad \text{(B.78)}
$$

Plugging this result into the bound from above yields

$$
\begin{aligned}
&\left\| \partial_\theta \left[\log_{K_1, M_1} \sigma_v - \log_{K_1, M_1} \tilde{\sigma}_v \right] \right\| \\
&\leq \frac{M_1 \|a\|_1 \pi}{2} \left(e 2^{\mu+1+n_h} \lambda_{max}^{\mu+1} \left\| \mathrm{Tr}_h \left[e^{-H} \right] \right\| \max_p \left(\frac{\mu+1-p}{eZ} \right)^{\mu+1-p} \left(\frac{\Delta}{\mu-1} \right)^\mu \frac{1}{\mu+1} \right) \\
&\quad + \frac{M_1 \|a\|_1 \pi}{2} \left(\mu \left\| \mathcal{L}'_{\mu,j}(\theta_j) \right\|_\infty \varepsilon_2 \right),
\end{aligned}
\tag{B.79}
$$

Note that under the reasonable assumption that $2 \leq \mu \ll Z$, the maximum is achieved for $p = \mu + 1$, and we hence obtain the upper bound

$$
\begin{aligned}
&\frac{M_1 \|a\|_1 \pi}{2} \left(2^{n_h} e (2|\lambda_{max}|)^{\mu+1} \left\| \mathrm{Tr}_h \left[e^{-H} \right] \right\| \left(\frac{\Delta}{\mu-1} \right)^\mu \frac{1}{\mu+1} + \mu \left\| \mathcal{L}'_{\mu,j}(\theta_j) \right\|_\infty \varepsilon_2 \right) \\
&\leq \frac{M_1 \|a\|_1 \pi}{2} \left(2^{n_h} e (2|\lambda_{max}|)^{\mu+1} \left\| \mathrm{Tr}_h \left[e^{-H} \right] \right\| \left(\frac{\Delta}{\mu-1} \right)^\mu + \mu \left\| \mathcal{L}'_{\mu,j}(\theta_j) \right\|_\infty \varepsilon_2 \right),
\end{aligned}
\tag{B.80}
$$

and we can hence obtain a bound on μ, the grid point number, in order to achieve an error of $\varepsilon/6 > 0$ for the former term, which is given by

$$
\mu \geq (|\lambda_{max}|\Delta) \exp \left(W \left(\frac{\log \left(2^{n_h} \frac{6 M_1 \|a\|_1 e^2 |\lambda_{max}| \pi \left\| \mathrm{Tr}_h \left[e^{-H} \right] \right\|}{\varepsilon} \right)}{2 \lambda_{max} \Delta} \right) \right),
\tag{B.81}
$$

where W is the Lambert function, also known as product-log function, which generally grows slower than the logarithm in the asymptotic limit. Note that μ can hence be lower bounded by

$$
\mu \geq n_h + \log \left(\frac{6 M_1 \|a\|_1 e^2 |\lambda_{max}| \pi \left\| \mathrm{Tr}_h \left[e^{-H} \right] \right\|}{\varepsilon} \right) := n_h + \log \left(\frac{M_1 \Lambda}{\varepsilon} \right).
\tag{B.82}
$$

For convenience, let us choose ε such that $n_h + \log(M_1 \Lambda / \varepsilon)$ is an integer. We do this simply to avoid having to keep track of ceiling or floor functions in the following discussion where we will choose $\mu = n_h + \log(M_1 \Lambda / \varepsilon)$.

For the second part, we will bound the derivative of the Lagrangian interpo-

lation polynomial. First, note that $\mathscr{L}'_{\mu,j}(\boldsymbol{\theta}) = \sum_{l=0;l\neq j}^{\mu} \left(\prod_{k=0;k\neq j,l} \frac{\theta-\theta_k}{\theta_j-\theta_k}\right) \frac{1}{\theta_j-\theta_l}$ for a chosen discretization of the space such that $\theta_k - \theta_j = (k-j)\Delta/\mu$ can be bound by using a central difference formula, such that we use an uneven number of points (i.e. we take $\mu = 2\kappa + 1$ for positive integer κ) and chose the point m at which we evaluate the gradient as the central point of the mesh. Note that in this case he have that for $\mu \geq 5$ and θ_m being the parameters at the midpoint of the stencil

$$
\begin{aligned}
\left\|\mathscr{L}'_{\mu,j}\right\|_\infty &\leq \sum_{l\neq j}\prod_{k\neq j,l}\frac{|\theta_m-\theta_k|}{|\theta_j-\theta_k|}\frac{1}{|\theta_l-\theta_j|} \leq \frac{(\kappa!)^2}{(\kappa!)^2}\frac{\mu}{\Delta}\sum_{l\neq j}\frac{1}{|l-j|} \\
&\leq \frac{2\mu}{\Delta}\sum_{l=1}^{\kappa}\frac{1}{l} \\
&\leq \frac{2\mu}{\Delta}\left(1+\int_1^{\kappa-1}\frac{1}{\ell}d\ell\right) = \frac{2\mu}{\Delta}\left(1+\log((\mu-3)/2)\right) \leq \frac{5\mu}{\Delta}\log(\mu/2),
\end{aligned}
$$

(B.83)

where the last inequality follows from the fact that $\mu \geq 5$ and $1 + \ln(5/2) < (5/2)\ln(5/2)$. Now, plugging in the μ from (B.94), we find that this error is bound by

$$
\left\|\mathscr{L}'_{\mu,j}\right\|_\infty \leq \frac{5n_h + 5\log\left(\frac{M_1\Lambda}{\varepsilon}\right)}{\Delta}\log(n_h/2 + \log\left(\frac{M_1\Lambda}{\varepsilon}\right)/2) = \tilde{O}\left(\frac{n_h + \log\left(\frac{M_1\Lambda}{\varepsilon}\right)}{\Delta}\right),
$$

(B.84)

If we want an upper bound of $\varepsilon/6$ for the second term of the error in (B.80), we hence require

$$
\begin{aligned}
\varepsilon_2 &\leq \frac{\varepsilon}{15M_1\|a\|_1\pi\mu\left\|\mathscr{L}'_{\mu,j}(\theta_j)\right\|_\infty} \\
&\leq \frac{\varepsilon\Delta}{15M_1\|a\|_1\pi\left(n_h+\log(M_1\Lambda/\varepsilon)\right)^2\log((n_h/2)+\log(M_1\Lambda/\varepsilon)/2)} \\
&\leq \frac{\varepsilon\Delta}{15M_1\|a\|_1\pi\mu^2\log((\mu-1)/2)}
\end{aligned}
$$

(B.85)

We hence obtain that the approximation error due to the divided differences and Fourier series approximation of σ_v is bounded by $\varepsilon/3$ for the above choice of ε_2

and μ. This bounds the second term in (B.72) by $\varepsilon/3$.

Finally, we need to take into account the error $\left\| \partial_\theta [\log_{K_1,M_1} \tilde\sigma_v - \log^s_{K_1,M_1} \tilde\sigma_v] \right\|$ which we introduce through the sampling process, i.e., through the finite sample estimate of $\mathbb{E}_s[\cdot]$ here indicated with the superscript s over the logarithm. Note that this error can be bound straight forward by (B.57). We only need to bound the error introduced via the finite amount of samples we take, which is a well-known procedure. The concrete bounds for the sample error when estimating the expectation value are stated in the following lemma.

Lemma 22. *Let σ_m be the sample standard deviation of the random variable*

$$\tilde{\mathbb{E}}_{s\in[0,1]} \left[\mathrm{Tr} \left(\rho\, e^{\frac{is\pi m}{2} \sigma_v} e^{\frac{i\pi m'}{2} \sigma_v(\theta_j)} e^{\frac{i(1-s)\pi m}{2} \sigma_v} \right) \right], \tag{B.86}$$

such that the sample standard deviation is given by $\sigma_k = \frac{\sigma_m}{\sqrt{k}}$. Then with probability at least $1-\delta_s$, we can obtain an estimate which is within $\varepsilon_s \sigma_m$ of the mean by taking $k = \frac{4}{\varepsilon_s^2}$ samples for each sample estimate and taking the median of $O(\log(1/\delta_s))$ such samples.

Proof. From Chebyshev's inequality taking $k = \frac{4}{\varepsilon_s^2}$ samples implies that with probability of at least $p = 3/4$ each of the mean estimates is within $2\sigma_k = \varepsilon_s \sigma_m$ from the true mean. Therefore, using standard techniques, we take the median of $O(\log(1/\delta_s))$ such estimates which gives us with probability $1-\delta_s$ an estimate of the mean with error at most $\varepsilon_s \sigma_m$, which implies that we need to repeat the procedure $O\left(\frac{1}{\varepsilon_s^2} \log\left(\frac{1}{\delta_s} \right) \right)$ times. $\qquad\square$

We can then bound the error of the sampling step in the final estimate, denoting with ε_s the sample error, as

$$\sum_{m=-M_1}^{M_1} \sum_{m'=-M_2}^{M_2} \left| \frac{ic_m c_{m'} m\pi}{2} \right| \sum_{j=0}^{\mu} |\mathscr{L}'_{\mu,j}(\theta)| \varepsilon_s \sigma_m$$

$$\leq \frac{5\|a\|_1 M_1 \varepsilon_s \sigma_m \pi \mu^2 \log\left(\frac{\mu}{2}\right)}{\Delta} \leq \frac{\varepsilon}{3}, \tag{B.87}$$

We hence find that for

$$\varepsilon_s \leq \frac{\varepsilon\Delta}{15\|a\|_1 M_1 \sigma_m \pi \mu^2 \log\left(\frac{\mu}{2}\right)}$$

$$\leq \frac{\varepsilon\Delta}{15 M_1 \|a\|_1 \sigma_m \pi \left(n_h + \log(M_1\Lambda/\varepsilon)\right)^2 \log((n_n/2) + \log(M_1\Lambda/\varepsilon)/2)}$$

$$\leq \frac{\varepsilon\Delta}{15 M_1 \|a\|_1 \sigma_m \pi \mu^2 \log(\mu/2)} \tag{B.88}$$

also the last term in (B.58) can be bounded by $\varepsilon/3$, which together results in an overall error of ε for the various approximation steps, which concludes the proof.

□

Notably all quantities which occure in our bounds are only polynomial in the number of the qubits. The lower bounds for the choice of parameters are summa-

rized in the following.

$$M_1 \geq \sqrt{L \log \left(\frac{9 \left\| \frac{\partial \sigma_v}{\partial \theta} \right\| K_1 L \pi}{2\varepsilon} \right)} \tag{B.89}$$

$$M_2 \geq \left\lceil \log \left(\frac{4}{\varepsilon_2} \right) \sqrt{(2 \log \delta_u^{-1})^{-1}} \right\rceil \tag{B.90}$$

$$K_1 \geq \log((1 - \|\sigma_v\|)\varepsilon/9) / \log(\|\sigma_v\|) \tag{B.91}$$

$$K_2 \geq \frac{\log(4/\varepsilon_2)}{\log(\delta_u^{-1})} \tag{B.92}$$

$$L \geq \frac{\log \left(\frac{\varepsilon}{9\pi K_1 \left\| \frac{\partial \sigma_v}{\partial \theta} \right\|} \right)}{\log(\|\sigma_v\| \pi)} \tag{B.93}$$

$$\mu \geq n_h + \log \left(\frac{6M_1 \|a\|_1 e^2 |\lambda_{max}| \pi \left\| \mathrm{Tr}_h \left[e^{-H} \right] \right\|}{\varepsilon} \right) := n_h + \log(M_1 \Lambda/\varepsilon) \tag{B.94}$$

$$\begin{aligned} \varepsilon_2 &\leq \frac{\varepsilon \Delta}{15M_1 \|a\|_1 \pi (n_h + \log(M_1\Lambda/\varepsilon))^2 \log((n_h/2) + \log(M_1\Lambda/\varepsilon)/2)} \\ &\leq \frac{\varepsilon \Delta}{15M_1 \|a\|_1 \pi \mu^2 \log((\mu - 1)/2)} \end{aligned} \tag{B.95}$$

$$\begin{aligned} \varepsilon_s &\leq \frac{\varepsilon \Delta}{15M_1 \|a\|_1 \sigma_m \pi (n_h + \log(M_1\Lambda/\varepsilon))^2 \log((n_n/2) + \log(M_1\Lambda/\varepsilon)/2)} \\ &\leq \frac{\varepsilon \Delta}{15M_1 \|a\|_1 \sigma_m \pi \mu^2 \log(\mu/2)} \end{aligned} \tag{B.96}$$

B.3.1 Operationalising

In the following we will make use of two established subroutines, namely sample based Hamiltonian simulation (aka the LMR protocol) [13], as well as the Hadamard test, in order to evaluate the gradient approximation as defined in (B.57). In order to hence derive the query complexity for this algorithm, we only need to multiply the cost of the number of factors we need to evaluate with the query complexity of these routines. For this we will rely on the following result.

Theorem 17 (Sample based Hamiltonian simulation [57])**.** *Let $0 \leq \varepsilon_h \leq 1/6$ be an error parameter and let ρ be a density for which we can obtain multiple copies through queries to a oracle O_ρ. We can then simulate the time evolution $e^{-i\rho t}$ up*

to error ε_h in trace norm as long as $\varepsilon_h/t \leq 1/(6\pi)$ with $\Theta(t^2/\varepsilon_h)$ copies of ρ and hence queries to O_ρ.

We in particularly need to evaluate terms of the form

$$\text{Tr}\left(\rho e^{\frac{is\pi m}{2}\sigma_v} e^{\frac{i\pi m'}{2}\sigma_v(\theta_j)} e^{\frac{i(1-s)\pi m}{2}\sigma_v}\right) \tag{B.97}$$

Note that we can simulate every term in the trace (except ρ) via the sample based Hamiltonian simulation approach to error ε_h in trace norm. This will introduce a additional error which we need to take into account for the analysis. Let $\tilde{U}_i, i \in \{1,2,3\}$ be the unitaries such that $\left\|U_i - \tilde{U}_i\right\|_* \leq \varepsilon_h$ where the U_i are corresponding to the factors in (B.97), i.e., $U_1 := e^{\frac{is\pi m}{2}\sigma_v}$, $U_2 := e^{\frac{i\pi m'}{2}\sigma_v(\theta_j)}$, and $U_3 := e^{\frac{i(1-s)\pi m}{2}\sigma_v}$. We can then bound the error as follows. First note that $\left\|\tilde{U}_i\right\| \leq \left\|\tilde{U}_i - U_i\right\| + \left\|U_i\right\| \leq 1 + \varepsilon_h$, using Theorem 17 and the fact that the spectral norm is upper bounded by the trace norm.

$$\begin{aligned}
\text{Tr}\left(\rho U_1 U_2 U_3\right) - \text{Tr}\left(\rho \tilde{U}_1 \tilde{U}_2 \tilde{U}_3\right) \leq \\
&= \text{Tr}\left(\rho U_1 U_2 U_3 - \rho \tilde{U}_1 \tilde{U}_2 \tilde{U}_3\right) \\
&\leq \left\|U_1 U_2 U_3 - \tilde{U}_1 \tilde{U}_2 \tilde{U}_3\right\| \\
&\leq \left\|U_1 - \tilde{U}_1\right\| \left\|\tilde{U}_2\right\| \left\|\tilde{U}_3\right\| + \left\|U_2 - \tilde{U}_2\right\| \tilde{U}_3 + \left\|U_3 - \tilde{U}_3\right\| \\
&\leq \left\|U_1 - \tilde{U}_1\right\|_* (1 + \varepsilon_h)^2 + \left\|U_2 - \tilde{U}_2\right\|_* (1 + \varepsilon_h) + \left\|U_3 - \tilde{U}_3\right\|_* \\
&\leq \varepsilon_h (1 + \varepsilon_h)^2 + \varepsilon_h (1 + \varepsilon_h) + \varepsilon_h = O(\varepsilon_h),
\end{aligned} \tag{B.98}$$

neglecting higher orders of ε_h, and where in the first step we applied the Von-Neumann trace inequality and the fact that ρ is Hermitian, and in the last step we used the results of Theorem 17. We hence require $O((\max\{M_1, M_2\}\pi)^2/\varepsilon_h)$ queries to the oracles for σ_v for the evaluation of each term in the multi sum in (B.57). Note that the Hadamard test has a query cost of $O(1)$. In order to hence achieve an overall error of ε in the gradient estimation we require the error introduced by the sample based Hamiltonian simulation also to be of $O(\varepsilon)$. In order to do so we require $\varepsilon_h \leq O(\frac{\varepsilon\Delta}{5\|a\|_1 M_1 \pi \mu^2 \log(\mu/2)})$, similar to the sample based error which yield the query

complexity of

$$O\left(\frac{\max\{M_1,M_2\}^2\,\|a\|_1\,M_1\pi^3\mu^2\log(\mu/2)}{\varepsilon\Delta}\right) \tag{B.99}$$

Adjusting the constants gives then the required bound of ε of the total error and the query complexity for the algorithm to the Gibbs state preparation procedure is consequentially given by the number of terms in (B.57) times the query complexity for the individual term, yielding

$$O\left(\frac{M_1^2 M_2 \max\{M_1,M_2\}^2\,\|a\|_1\,\sigma_m\pi^3\mu^3\log\left(\frac{\mu}{2}\right)}{\varepsilon\,\varepsilon_s^2\Delta}\right), \tag{B.100}$$

and classical precomputation polynomial in $M_1,M_2,K_1,L,s,\Delta,\mu$, where the different quantities are defines in eq. (B.89-B.96).

Taking into account the query complexity of the individual steps then results in Theorem 13. We proceed by proving this theorem next.

Proof of Theorem 13. The runtime follows straight forward by using the bounds derived in (B.99) and Lemma 21, and by using the bounds for the parameters $M_1,M_2,K_1,L,\mu,\Delta,s$ given in eq. (B.89-B.96). For the success probability for estimating the whole gradient with dimensionality d, we can now again make use of the boosting scheme used in (B.23) to be

$$\tilde{O}\left(\frac{d\,\|a\|_1^3\,\sigma_m^3\mu^5\log^3(\mu/2)\mathrm{polylog}\left(\frac{\left\|\frac{\partial\sigma_v}{\partial\theta}\right\|}{\varepsilon},\frac{n_h^2\|a\|_1\sigma_m}{\varepsilon\Delta}\right)}{\varepsilon^3\Delta^3}\log(d)\right), \tag{B.101}$$

where $\mu = n_h + \log(M_1\Lambda/\varepsilon)$.

Next we need to take into account the errors from the Gibbs state preparation given in Lemma 17. For this note that the error between the perfect Hamiltonian simulation of σ_v and the sample based Hamiltonian simulation with an erroneous density matrix denoted by $\tilde{U}$, i.e., including the error from the Gibbs state preparation pro-

cedure, is given by

$$\left\| \tilde{U} - e^{-i\sigma_v t} \right\| \leq \left\| \tilde{U} - e^{-i\tilde{\sigma}_v t} \right\| + \left\| e^{-i\tilde{\sigma}_v t} - e^{-i\sigma_v t} \right\|$$

$$\leq \varepsilon_h + \varepsilon_G t \tag{B.102}$$

where ε_h is the error of the sample based Hamiltonian simulation, which holds since the trace norm is an upper bound for the spectral norm, and $\|\sigma_v - \tilde{\sigma}_v\| \leq \varepsilon_G$ is the error for the Gibbs state preparation from Theorem 15 for a d-sparse Hamiltonian, for a cost

$$\tilde{\mathcal{O}} \left(\sqrt{\frac{N}{z}} \|H\| d \log \left(\frac{\|H\|}{\varepsilon_G} \right) \log \left(\frac{1}{\varepsilon_G} \right) \right).$$

From (B.98) we know that the error ε_h propagates nearly linear, and hence it suffices for us to take $\varepsilon_G \leq \varepsilon_h / t$ where $t = O(\max\{M_1, M_2\})$ and adjust the constants $\varepsilon_h \leftarrow \varepsilon_h / 2$ in order to achieve the same precision ε in the final result. We hence require

$$\tilde{\mathcal{O}} \left(\sqrt{\frac{N}{z}} \|H(\theta)\| \log \left(\frac{\|H(\theta)\| \max\{M_1, M_2\}}{\varepsilon_h} \right) \log \left(\frac{\max\{M_1, M_2\}}{\varepsilon_h} \right) \right) \tag{B.103}$$

and using the ε_h from before we hence find that this s bound by

$$\tilde{\mathcal{O}} \left(\sqrt{\frac{N}{z}} \|H(\theta)\| \log \left(\frac{\|H(\theta)\| n_h^2}{\varepsilon \Delta} \right) \log \left(\frac{n_h^2}{\varepsilon \Delta} \right) \right) \tag{B.104}$$

query complexity to the oracle of H for the Gibbs state preparation.

The procedure succeeds with probability at least $1 - \delta_s$ for a single repetition for each entry of the gradient. In order to have a failure probability of the final algorithm of less than $1/3$, we need to repeat the procedure for all D dimensions of the gradient and take for each the median over a number of samples. Let n_f be as previously the number of instances of the one component of the gradient such that the error is larger than $\varepsilon_s \sigma_m$ and n_s be the number of instances with an error $\leq \varepsilon_s \sigma_m$, and the result that we take is the median of the estimates, where we take $n = n_s + n_f$ samples. The algorithm gives a wrong answer for each dimension if $n_s \leq \left\lfloor \frac{n}{2} \right\rfloor$, since then the median is a sample such that the error is larger than $\varepsilon_s \sigma_m$. Let $p = 1 - \delta_s$

be the success probability to draw a positive sample, as is the case of our algorithm. Since each instance of (recall that each sample here consists of a number of samples itself) from the algorithm will independently return an estimate for the entry of the gradient, the total failure probability is bounded by the union bound, i.e.,

$$\Pr_{fail} \le D \cdot \Pr\left[n_s \le \left\lfloor \frac{n}{2} \right\rfloor \right] \le D \cdot e^{-\frac{n}{2(1-\delta_s)}\left((1-\delta_s)-\frac{1}{2}\right)^2} \le \frac{1}{3}, \tag{B.105}$$

which follows from the Chernoff inequality for a binomial variable with $1 - \delta_s > 1/2$, which is given in our case for a proper choice of $\delta_s < 1/2$. Therefore, by taking $n \ge \frac{2-2\delta_s}{(1/2-\delta_s)^2} \log(3D) = O(\log(3D))$, we achieve a total failure probability of at least $1/3$ for a constant, fixed δ_s. Note that this hence results in an multiplicative factor of $O(\log(D))$ in the query complexity of (5.25).

The total query complexity to the oracle O_ρ for a purified density matrix of the data ρ and the Hamiltonian oracle O_H is then given by

$$\tilde{O}\left(\sqrt{\frac{N}{z}} \frac{d\log(d) \|H(\theta)\| \|a\|_1^3 \sigma_m^3 \mu^5 \log^3(\mu/2) \text{polylog}\left(\frac{\left\| \frac{\partial \sigma_v}{\partial \theta} \right\|}{\varepsilon}, \frac{n_h^2 \|a\|_1 \sigma_m}{\varepsilon \Delta}, \|H(\theta)\| \right)}{\varepsilon^3 \Delta^3} \right), \tag{B.106}$$

which reduces to

$$\tilde{O}\left(\sqrt{\frac{N}{z}} \frac{D \|H(\theta)\| d\mu^5 \alpha}{\varepsilon^3} \right), \tag{B.107}$$

hiding the logarithmic factors in the $\tilde{O}$ notation. $\qquad\square$

Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[3] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[4] Igor L Markov. Limits on fundamental limits to computation. *Nature*, 512(7513):147–154, 2014.

[5] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209):20170551, 2018.

[6] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.

[7] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Architectures for a quantum random access memory. *Physical Review A*, 78(5):052310, 2008.

[8] Seth Lloyd. Universal quantum simulators. *Science*, pages 1073–1078, 1996.

[9] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. Quantum deep learning. *arXiv preprint arXiv:1412.3489*, 2014.

[10] Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. Falkon: An optimal large scale kernel method. In *Advances in Neural Information Processing Systems*, pages 3888–3898, 2017.

[11] B David Clader, Bryan C Jacobs, and Chad R Sprouse. Preconditioned quantum linear system algorithm. *Physical review letters*, 110(25):250504, 2013.

[12] Aram Harrow and Rolando P. La Placa. Limitations of quantum preconditioning using a sparse approximate inverse. *Unpublished work*, 2017.

[13] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631, 2014.

[14] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.

[15] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 217–228, 2019.

[16] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[17] Ewin Tang. Quantum-inspired classical algorithms for principal component analysis and supervised clustering. *arXiv preprint arXiv:1811.00414*, 2018.

[18] Nai-Hui Chia, Han-Hsuan Lin, and Chunhao Wang. Quantum-inspired sublinear classical algorithms for solving low-rank linear systems. *arXiv preprint arXiv:1811.04852*, 2018.

[19] András Gilyén, Seth Lloyd, and Ewin Tang. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. *arXiv preprint arXiv:1811.04909*, 2018.

[20] Nai-Hui Chia, Tongyang Li, Han-Hsuan Lin, and Chunhao Wang. Quantum-inspired classical sublinear-time algorithm for solving low-rank semidefinite programming via sampling approaches. *arXiv preprint arXiv:1901.03254*, 2019.

[21] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 387–400, 2020.

[22] Danial Dervovic, Mark Herbster, Peter Mountney, Simone Severini, Naïri Usher, and Leonard Wossnig. Quantum linear systems algorithms: a primer. *arXiv preprint arXiv:1802.08227*, 2018.

[23] Chunhao Wang and Leonard Wossnig. A quantum algorithm for simulating non-sparse hamiltonians. *Quantum Information & Computation*, 20(7&8):597–615, 2020.

[24] Alessandro Rudi, Leonard Wossnig, Carlo Ciliberto, Andrea Rocchetto, Massimiliano Pontil, and Simone Severini. Approximating hamiltonian dynamics with the nyström method. *Quantum*, 4:234, 2020.

[25] Carlo Ciliberto, Andrea Rocchetto, Alessandro Rudi, and Leonard Wossnig. Fast quantum learning with statistical guarantees. *arXiv preprint arXiv:2001.10477*, 2020.

[26] Nathan Wiebe and Leonard Wossnig. Generative training of quantum boltzmann machines with hidden units. *arXiv preprint arXiv:1905.09902*, 2019.

[27] Hongxiang Chen, Leonard Wossnig, Simone Severini, Hartmut Neven, and Masoud Mohseni. Universal discriminative quantum neural networks. *arXiv preprint arXiv:1805.08654*, 2018.

[28] Marcello Benedetti, Edward Grant, Leonard Wossnig, and Simone Severini. Adversarial quantum circuit learning for pure state approximation. *New Journal of Physics*, 21(4):043023, 2019.

[29] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019.

[30] Shuxiang Cao, Leonard Wossnig, Brian Vlastakis, Peter Leek, and Edward Grant. Cost-function embedding and dataset encoding for machine learning with parametrized quantum circuits. *Physical Review A*, 101(5):052309, 2020.

[31] I Rungger, N Fitzpatrick, H Chen, CH Alderete, H Apel, A Cowtan, A Patterson, D Munoz Ramo, Y Zhu, NH Nguyen, et al. Dynamical mean field theory algorithm and experiment on quantum computers. *arXiv preprint arXiv:1910.04735*, 2019.

[32] Andrew Patterson, Hongxiang Chen, Leonard Wossnig, Simone Severini, Dan Browne, and Ivan Rungger. Quantum state discrimination using noisy quantum neural networks. *arXiv preprint arXiv:1911.00352*, 2019.

[33] Jules Tilly, Glenn Jones, Hongxiang Chen, Leonard Wossnig, and Edward Grant. Computation of molecular excited states on ibmq using a discriminative variational quantum eigensolver. *arXiv preprint arXiv:2001.04941*, 2020.

[34] Howard E Haber. Notes on the matrix exponential and logarithm. 2018.

[35] Nicholas J Higham. *Functions of matrices: theory and computation*, volume 104. Siam, 2008.

[36] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[37] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: Improved regression techniques via faster Hamiltonian simulation. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132, pages 33:1–33:14, 2019.

[38] Nathan Wiebe, Daniel Braun, and Seth Lloyd. Quantum algorithm for data fitting. *Physical review letters*, 109(5):050505, 2012.

[39] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. Prediction by linear regression on a quantum computer. *Physical Review A*, 94(2):022342, 2016.

[40] Guoming Wang. Quantum algorithm for linear regression. *Physical review A*, 96(1):012335, 2017.

[41] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. In *Innovations in Theoretical Computer Science (ITCS'17)*, 2017.

[42] Dan-Bo Zhang, Shi-Liang Zhu, and ZD Wang. Nonlinear regression based on a hybrid quantum computer. *arXiv preprint arXiv:1808.09607*, 2018.

[43] Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291–293, 2015.

[44] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[45] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[46] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[47] Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39(1):1–49, 2002.

[48] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[49] Carl Rasmussen and Chris Williams. Gaussian processes for machine learning. *Gaussian Processes for Machine Learning*, 2006.

[50] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[51] Yuchen Zhang, John Duchi, and Martin Wainwright. Divide and conquer kernel ridge regression. In *Conference on Learning Theory*, pages 592–617, 2013.

[52] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5, 2007.

[53] Alex J Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. pages 911–918. Morgan Kaufmann, 2000.

[54] Christopher KI Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

[55] Alessandro Rudi, Raffaello Camoriano, and Lorenzo Rosasco. Less is more: Nyström computational regularization. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1657–1665, Cambridge, MA, USA, 2015. MIT Press.

[56] Alessandro Rudi, Leonard Wossnig, Carlo Ciliberto, Andrea Rocchetto, Massimiliano Pontil, and Simone Severini. Approximating Hamiltonian dynamics with the Nyström method. *Quantum*, 4:234, 2020.

[57] Shelby Kimmel, Cedric Yen-Yu Lin, Guang Hao Low, Maris Ozols, and Theodore J Yoder. Hamiltonian simulation with optimal sample complexity. *npj Quantum Information*, 3(1):13, 2017.

[58] Tongyang Li, Shouvanik Chakrabarti, and Xiaodi Wu. Sublinear quantum algorithms for training linear and kernel-based classifiers. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3815–3824, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[59] L Lo Gerfo, Lorenzo Rosasco, Francesca Odone, E De Vito, and Alessandro Verri. Spectral algorithms for supervised learning. *Neural Computation*, 20(7):1873–1897, 2008.

[60] Harry Hochstadt. *Integral equations*, volume 91. John Wiley & Sons, 2011.

[61] Joel A Tropp et al. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015.

[62] Alex B Grilo and Iordanis Kerenidis. Learning with errors is easy with quantum samples. *arXiv preprint arXiv:1702.08255*, 2017.

[63] Varun Kanade, Andrea Rocchetto, and Simone Severini. Learning dnfs under product distributions via {\mu}-biased quantum fourier sampling. *arXiv preprint arXiv:1802.05690*, 2018.

[64] Yu Tong, Dong An, Nathan Wiebe, and Lin Lin. Fast inversion, preconditioned quantum linear system solvers, and fast evaluation of matrix functions. *arXiv preprint arXiv:2008.13295*, 2020.

[65] Siyuan Ma and Mikhail Belkin. Diving into the shallows: a computational perspective on large-scale shallow learning. In *Advances in Neural Information Processing Systems*, pages 3778–3787, 2017.

[66] Alon Gonen, Francesco Orabona, and Shai Shalev-Shwartz. Solving ridge regression using sketched preconditioned svrg. In *International Conference on Machine Learning*, pages 1397–1405, 2016.

[67] Haim Avron, Kenneth L Clarkson, and David P Woodruff. Faster kernel ridge regression using sketching and preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1116–1138, 2017.

[68] Gregory E Fasshauer and Michael J McCourt. Stable evaluation of gaussian radial basis function interpolants. *SIAM Journal on Scientific Computing*, 34(2):A737–A762, 2012.

[69] David P Woodruff. Sketching as a tool for numerical linear algebra. *arXiv preprint arXiv:1411.4357*, 2014.

[70] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

[71] Michael W Mahoney. Lecture notes on randomized linear algebra. *arXiv preprint arXiv:1608.04481*, 2016.

[72] Jeremy Adcock, Euan Allen, Matthew Day, Stefan Frick, Janna Hinchliff, Mack Johnson, Sam Morley-Short, Sam Pallister, Alasdair Price, and Stasja Stanisic. Advances in quantum machine learning. *arXiv preprint arXiv:1512.02900*, 2015.

[73] Damian S. Steiger and Matthias Troyer. Racing in parallel: Quantum versus classical. In *Quantum Machine Learning Workshop*, Waterloo, CA, August 2016. Perimeter Institute for theoretical Physics.

[74] Oded Regev and Liron Schiff. Impossibility of a quantum speed-up with a faulty oracle. In *International Colloquium on Automata, Languages, and Programming*, pages 773–781. Springer, 2008.

[75] Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan. On the robustness of bucket brigade quantum ram. *New Journal of Physics*, 17(12):123010, 2015.

[76] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.

[77] Richard P Feynman. Quantum mechanical computers. *Foundations of physics*, 16(6):507–531, 1986.

[78] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

[79] Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC 2003)*, pages 20–29. ACM, 2003.

[80] Dominic W Berry, Graeme Ahokas, Richard Cleve, and Barry C Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007.

[81] Dominic W Berry and Andrew M Childs. Black-box Hamiltonian simulation and unitary implementation. *Quantum Information & Computation*, 12(1-2):29–62, 2012.

[82] Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma. Exponential improvement in precision for simulating sparse Hamiltonians. *Forum of Mathematics, Sigma*, 5, 2017.

[83] Andrew M Childs. On the relationship between continuous-and discrete-time quantum walk. *Communications in Mathematical Physics*, 294(2):581–603, 2010.

[84] Andrew M Childs and Nathan Wiebe. Hamiltonian simulation using linear combinations of unitary operations. *Quantum Information & Computation*, 12(11-12):901–924, 2012.

[85] David Poulin, Angie Qarry, Rolando Somma, and Frank Verstraete. Quantum simulation of time-dependent Hamiltonians and the convenient illusion of hilbert space. *Physical Review Letters*, 106(17):170501, 2011.

[86] Nathan Wiebe, Dominic W Berry, Peter Høyer, and Barry C Sanders. Simulating quantum dynamics on a quantum computer. *Journal of Physics A: Mathematical and Theoretical*, 44(44):445308, 2011.

[87] Guang Hao Low and Isaac L Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, 2019.

[88] Dominic W Berry and Leonardo Novo. Corrected quantum walk for optimal Hamiltonian simulation. *Quantum Information & Computation*, 16(15-16):1295–1317, 2016.

[89] Guang Hao Low and Isaac L Chuang. Hamiltonian simulation by uniform spectral amplification. *arXiv preprint arXiv:1707.05391*, 2017.

[90] Dominic W Berry, Andrew M Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science (FOCS 2015)*, pages 792–809. IEEE, 2015.

[91] Guang Hao Low and Isaac L Chuang. Optimal Hamiltonian simulation by quantum signal processing. *Physical Review Letters*, 118(1):010501, 2017.

[92] Guang Hao Low. Hamiltonian simulation with nearly optimal dependence on spectral norm. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC 2019)*, pages 491–502, 2019.

[93] Andrew M Childs, Robin Kothari, and Rolando D Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017.

[94] Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. Quantum linear system algorithm for dense matrices. *Physical Review Letters*, 120(5):050502, 2018.

[95] Andrew M. Childs and Robin Kothari. Limitations on the simulation of non-sparse Hamiltonians. *Quantum Information & Computation*, 10(7&8):669–684, 2010.

[96] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 1990.

[97] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC 2019)*, pages 193–204. ACM, 2019.

[98] Nicholas J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM J. Matrix Anal. Appl.*, 26(4):1179–1193, April 2005.

[99] Nicholas J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM Rev.*, 51(4):747–764, November 2009.

[100] Awad H Al-Mohy and Nicholas J Higham. A new scaling and squaring algorithm for the matrix exponential. *SIAM Journal on Matrix Analysis and Applications*, 31(3):970–989, 2009.

[101] Awad H Al-Mohy and Nicholas J Higham. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM journal on scientific computing*, 33(2):488–511, 2011.

[102] Petros Drineas, Michael W. Mahoney, S. Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *Numer. Math.*, 117(2):219–249, February 2011.

[103] Lorenzo Orecchia, Sushant Sachdeva, and Nisheeth K. Vishnoi. Approximating the exponential, the lanczos method and an $\tilde{O}(m)$-time spectral algorithm for balanced separator. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, page 1141–1160, New York, NY, USA, 2012. Association for Computing Machinery.

[104] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, page 81–90, New York, NY, USA, 2004. Association for Computing Machinery.

[105] Daniel A Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

[106] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on computing*, 36(1):158–183, 2006.

[107] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nyström method. *Journal of Machine Learning Research*, 13(Apr):981–1006, 2012.

[108] Petros Drineas and Michael W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *J. Mach. Learn. Res.*, 6:2153–2175, December 2005.

[109] CKI. Williams, CE. Rasmussen, A. Schwaighofer, and V. Tresp. Observations on the nyström method for gaussian process prediction. 2002.

[110] Evert J Nyström. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54(1):185–204, 1930.

[111] Kai Zhang and James T Kwok. Clustered nyström method for large scale manifold learning and dimension reduction. *IEEE Transactions on Neural Networks*, 21(10):1576–1587, 2010.

[112] Ameet Talwalkar, Sanjiv Kumar, and Henry Rowley. Large-scale manifold learning. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[113] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nyström method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):214–225, January 2004.

[114] Mohamed-Ali Belabbas and Patrick J Wolfe. Fast low-rank approximation for covariance matrices. In *Computational Advances in Multi-Sensor Adaptive Processing, 2007. CAMPSAP 2007. 2nd IEEE International Workshop on*, pages 293–296. IEEE, 2007.

[115] Mohamed-Ali Belabbas and Patrick J Wolfe. On sparse representations of linear operators and the approximation of matrix products. In *Information Sciences and Systems, 2008. CISS 2008. 42nd Annual Conference on*, pages 258–263. IEEE, 2008.

[116] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. On sampling-based approximate spectral decomposition. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 553–560, New York, NY, USA, 2009. Association for Computing Machinery.

[117] Mu Li, James T. Kwok, and Bao-Liang Lu. Making large-scale nyström approximation possible. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 631–638, Madison, WI, USA, 2010. Omnipress.

[118] Lester Mackey, Ameet Talwalkar, and Michael I. Jordan. Divide-and-conquer matrix factorization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 1134–1142, Red Hook, NY, USA, 2011. Curran Associates Inc.

[119] Kai Zhang, Ivor W. Tsang, and James T. Kwok. Improved nyström low-rank approximation and error analysis. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1232–1239, New York, NY, USA, 2008. Association for Computing Machinery.

[120] Ivan Kassal, James D Whitfield, Alejandro Perdomo-Ortiz, Man-Hong Yung, and Alán Aspuru-Guzik. Simulating chemistry using quantum computers. *Annual review of physical chemistry*, 62:185–207, 2011.

[121] Stephen P Jordan and Pawel Wocjan. Efficient quantum circuits for arbitrary sparse unitaries. *Physical Review A*, 80(6):062301, 2009.

[122] A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997.

[123] Mario Szegedy. Quantum speed-up of markov chain based algorithms. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 32–41. IEEE, 2004.

[124] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Courier Corporation, 1964.

[125] Robin Kothari. *Efficient algorithms in quantum query complexity*. PhD thesis, University of Waterloo, 2014.

[126] W. G. Bickley, L. J. Comrie, J. C. P. Miller, D. H. Sadler, and A. J. Thompson. *Bessel Functions: Part II. Functions of Positive Integer Order*. Cambridge University Press, 1960.

[127] F. W. J. Olver. Error analysis of miller's recurrence algorithm. *Mathematics of Computation*, 18(85):65–74, 1964.

[128] Cupjin Huang, Michael Newman, and Mario Szegedy. Explicit lower bounds on strong quantum simulation. *arXiv preprint arXiv:1804.10368*, 2018.

[129] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

[130] Ritsuo Nakamoto. A norm inequality for hermitian operators. *The American mathematical monthly*, 110(3):238, 2003.

[131] Roy Mathias. Approximation of matrix-valued functions. *SIAM journal on matrix analysis and applications*, 14(4):1061–1063, 1993.

[132] Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Found. Comput. Math.*, 12(4):389–434, August 2012.

[133] Alexei Borisovich Aleksandrov and Vladimir Vsevolodovich Peller. Operator lipschitz functions. *Russian Mathematical Surveys*, 71(4):605, 2016.

[134] Daniel Hsu. Weighted sampling of outer products. *arXiv preprint arXiv:1410.4429*, 2014.

[135] Tianbao Yang, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in neural information processing systems*, pages 476–484, 2012.

[136] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems*, pages 1313–1320, 2009.

[137] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[138] Shusen Wang and Zhihua Zhang. Improving cur matrix decomposition and the nyström approximation via adaptive sampling. *The Journal of Machine Learning Research*, 14(1):2729–2769, 2013.

[139] Shusen Wang, Zhihua Zhang, and Tong Zhang. Towards more efficient spsd matrix approximation and cur matrix decomposition. *The Journal of Machine Learning Research*, 17(1):7329–7377, 2016.

[140] Zhikuan Zhao, Jack K Fitzsimons, and Joseph F Fitzsimons. Quantum assisted gaussian process regression. *arXiv preprint arXiv:1512.03929*, 2015.

[141] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. Quantum algorithms for topological and geometric analysis of big data. *arXiv preprint arXiv:1408.3106*, 2014.

[142] Juan Miguel Arrazola, Alain Delgado, Bhaskar Roy Bardhan, and Seth Lloyd. Quantum-inspired algorithms in practice. *arXiv preprint arXiv:1905.10415*, 2019.

[143] Yogesh Dahiya, Dimitris Konomis, and David P Woodruff. An empirical evaluation of sketching for numerical linear algebra. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1292–1300, 2018.

[144] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195, 2017.

[145] Rocco A Servedio and Steven J Gortler. Equivalences and separations between quantum and classical learnability. *SIAM Journal on Computing*, 33(5):1067–1092, 2004.

[146] Srinivasan Arunachalam and Ronald De Wolf. Optimal quantum sample complexity of learning algorithms. *The Journal of Machine Learning Research*, 19(1):2879–2878, 2018.

[147] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

[148] Nathan Wiebe, Alex Bocharov, Paul Smolensky, Krysta Svore, and Matthias Troyer. Quantum language processing. *arXiv preprint arXiv:1902.05162*, 2019.

[149] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812, 2018.

[150] Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *arXiv preprint arXiv:1804.00633*, 2018.

[151] Mária Kieferová and Nathan Wiebe. Tomography and generative training with quantum boltzmann machines. *Phys. Rev. A*, 96:062327, Dec 2017.

[152] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical review letters*, 122(4):040504, 2019.

[153] Jonathan Romero, Jonathan P Olson, and Alan Aspuru-Guzik. Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology*, 2(4):045001, 2017.

[154] Anthony Chefles and Stephen M Barnett. Strategies and networks for state-dependent quantum cloning. *Physical Review A*, 60(1):136, 1999.

[155] Hilbert J Kappen. Learning quantum models from quantum or classical data. *arXiv preprint arXiv:1803.11278*, 2018.

[156] Mohammad H Amin, Evgeny Andriyash, Jason Rolfe, Bohdan Kulchytskyy, and Roger Melko. Quantum boltzmann machine. *Physical Review X*, 8(2):021050, 2018.

[157] Daniel Crawford, Anna Levit, Navid Ghadermarzy, Jaspreet S Oberoi, and Pooya Ronagh. Reinforcement learning using quantum boltzmann machines. *arXiv preprint arXiv:1612.05695*, 2016.

[158] Marcello Benedetti, John Realpe-Gómez, Rupak Biswas, and Alejandro Perdomo-Ortiz. Quantum-assisted learning of hardware-embedded probabilistic graphical models. *Physical Review X*, 7(4):041052, 2017.

[159] Nathan Wiebe, Ashish Kapoor, Christopher Granade, and Krysta M Svore. Quantum inspired training for boltzmann machines. *arXiv preprint arXiv:1507.02642*, 2015.

[160] Emile Aarts and Jan Korst. Simulated annealing and boltzmann machines. 1988.

[161] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[162] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM, 2008.

[163] Nicolas Le Roux and Yoshua Bengio. Representational power of restricted boltzmann machines and deep belief networks. *Neural computation*, 20(6):1631–1649, 2008.

[164] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455, 2009.

[165] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 693–700, 2010.

[166] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical

representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.

[167] Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.

[168] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

[169] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE transactions on audio, speech, and language processing*, 20(1):14–22, 2011.

[170] Nitish Srivastava and Ruslan R Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230, 2012.

[171] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[172] Giacomo Torlai and Roger G Melko. Learning thermodynamics with boltzmann machines. *Physical Review B*, 94(16):165134, 2016.

[173] Yusuke Nomura, Andrew S Darmawan, Youhei Yamaji, and Masatoshi Imada. Restricted boltzmann machine learning for solving strongly correlated quantum systems. *Physical Review B*, 96(20):205152, 2017.

[174] Dorit Aharonov, Vaughan Jones, and Zeph Landau. A polynomial quantum algorithm for approximating the jones polynomial. *Algorithmica*, 55(3):395–421, 2009.

[175] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum sdp-solvers: Better upper and lower bounds. *arXiv preprint arXiv:1705.01843*, 2017.

[176] Scott Aaronson. The learnability of quantum states. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 463, pages 3089–3114. The Royal Society, 2007.

[177] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *arXiv preprint arXiv:2010.02174*, 2020.

[178] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R McClean. Power of data in quantum machine learning. *arXiv preprint arXiv:2011.01938*, 2020.

[179] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[180] LD Landau and EM Lifshitz. Statistical physics, vol. 5. *Course of theoretical physics*, 30, 1980.

[181] David Poulin and Pawel Wocjan. Sampling from the thermal quantum gibbs state and evaluating partition functions with a quantum computer. *Physical review letters*, 103(22):220502, 2009.

[182] Anirban Narayan Chowdhury and Rolando D Somma. Quantum algorithms for gibbs sampling and hitting-time estimation. *arXiv preprint arXiv:1603.02940*, 2016.

[183] Man-Hong Yung and Alán Aspuru-Guzik. A quantum–quantum metropolis algorithm. *Proceedings of the National Academy of Sciences*, 109(3):754–759, 2012.