Optimizing quantum optimization algorithms via faster quantum gradient computation

András Gilyén* Srinivasan Arunachalam* Nathan Wiebe†

Abstract

We consider a generic framework of optimization algorithms based on gradient descent. We develop a quantum algorithm that computes the gradient of a multi-variate real-valued function $f : \mathbb{R}^d \to \mathbb{R}$ by evaluating it at only a logarithmic number of times in superposition. Our algorithm is an improved version of Jordan's gradient computation algorithm [28], providing an approximation of the gradient ∇f with quadratically better dependence on the evaluation accuracy of f, for an important class of smooth functions. Furthermore, we show that objective functions arising from variational quantum circuits usually satisfy the necessary smoothness conditions, hence our algorithm provides a quadratic improvement in the complexity of computing their gradient. We also show that in a continuous phase-query model, our gradient computation algorithm has optimal query complexity up to poly-logarithmic factors, for a particular class of smooth functions. Moreover, we show that for low-degree multivariate polynomials our algorithm can provide exponential speedups compared to Jordan's algorithm in terms of the dimension d.

One of the technical challenges in applying our gradient computation procedure for quantum optimization problems is the need to convert between a probability oracle (which is common in quantum optimization procedures) and a phase oracle (which is common in quantum algorithms) of the objective function f. We provide *efficient* subroutines to perform this delicate interconversion between the two types of oracles incurring only a logarithmic overhead, which might be of independent interest. Finally, using these tools we improve the runtime of prior approaches for training quantum auto-encoders, variational quantum eigensolvers (VQE), and quantum approximate optimization algorithms (QAOA).

1 Introduction

Quantum optimization. Optimization is a fundamentally important task that touches on virtually every area of science. Recently, there have been many quantum algorithms that provide substantial improvements for several optimization problems [24, 18, 28, 26, 14, 10, 3, 9, 2, 31, 4, 13]. However, applying non-Grover techniques to real-word optimization problems has proven challenging, because generic problems often fail to satisfy the delicate requirements of advanced quantum techniques.

A different paradigm of quantum optimization is based on variational quantum circuits. These circuits are usually based on heuristics, nevertheless they are promising candidates for providing quantum advantage in real-word problems, including quantum simulation [39, 47], optimization [19], quantum neural networks [20] and machine learning [43]. These variational circuits usually try to optimize some objective function, which could correspond to, e.g., the energy of a quantum state in a molecule or the prediction loss in a learning model. These quantum circuits have the appealing feature that they often have low depth, therefore can potentially be implemented on NISQ (noisy intermediate-scale quantum) hardware. The training is usually performed by running the circuit several times and using classical gradient descent on the parameter space.

In the long term it is expected that training could become a bottleneck as the size of the variational quantum circuits grow, similarly to for example the training of *classical* deep neural networks. However, the ultimate limitations of variational training algorithms are not well understood. In particular it has been unclear whether it is possible to achieve improvements beyond the simple quantum speedups provided by amplitude estimation techniques. This underscores the importance of understanding the performance of training algorithms as we begin to push beyond NISQ-era devices. Our main contribution is showing that non-trivial speedups can be achieved via quantum gradient computation of the objective function. We remark that in this variational setting the prior works on quantum gradient descent [40, 30] are not applicable.

It is usually difficult to reduce the number of iterations using quantum computers in iterative algorithms such as gradient descent. We therefore focus on the cost per iteration, and speed up gradient computation. Since in general very little is known about the landscape

*QuSoft/CWI, Amsterdam, Netherlands. Supported by ERC Consolidator Grant QPROGRESS and partially supported by QuantERA project QuantAlgo 680-91-034. {gilyen,arunacha}@cwi.nl

†Station Q QuArC, Microsoft Research, USA. nawiebe@microsoft.com

of objective functions arising from variational quantum circuits, we study the problem in a black-box setting where we can only learn about the objective function by evaluating it at arbitrary points. This setting simplifies the problem and allows us to reason about upper and lower bounds on the cost of gradient computation.

Classical gradient-based optimization algorithms. We first give a high-level description of a basic classical gradient-based optimization technique. The problem is, given $p : \mathbb{R}^d \to \mathbb{R}$, compute

$$\text{OPT} = \min\{p(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^d\}. \tag{1.1}$$

A heuristic solution of the optimization problem (1.1) can be obtained by computing the *gradient* of p:

$$\nabla p = \left(\frac{\partial p}{\partial x_1}, \frac{\partial p}{\partial x_2}, \ldots, \frac{\partial p}{\partial x_d}\right) \tag{1.2}$$

It is a well-known fact in calculus that p decreases the *fastest* in the direction of $-(\nabla p(\boldsymbol{x}))$. This simple observation is the basis of gradient-based optimization algorithms. Given the generality of the optimization problem (1.1) and the simplicity of the algorithm, gradient-based techniques are widely used in mathematics, physics and engineering.

Probability oracles. Since quantum algorithms such as QAOA or VQE work with an objective function that needs to be learned by sampling, we assume the function is given by a *probability oracle*, which for every $\boldsymbol{x} \in \mathbb{R}^d$ acts as:

$$O_p\colon |\vec{0}\rangle|\boldsymbol{x}\rangle \mapsto \left(\sqrt{p(\boldsymbol{x})}|1\rangle|\psi_{\boldsymbol{x}}^{(1)}\rangle + \sqrt{1-p(\boldsymbol{x})}|0\rangle|\psi_{\boldsymbol{x}}^{(0)}\rangle\right)|\boldsymbol{x}\rangle \tag{1.3}$$

where the continuous variable $\boldsymbol{x}$ is represented binarily with some finite precision.[1]

The classical analogue of this model is when we get samples from the distribution $(p(\boldsymbol{x}), 1-p(\boldsymbol{x}))$. Using empirical estimation, $\mathcal{O}(1/\varepsilon^2)$ samples suffice for estimating $p(\boldsymbol{x})$ with precision $\mathcal{O}(\varepsilon)$. If the function is sufficiently smooth, using standard techniques we can compute an ε-approximation of $\nabla_i p(\boldsymbol{x})$ using a logarithmic number of such function estimations. Computing the gradient this way uses $\widetilde{\mathcal{O}}(d/\varepsilon^2)$ samples.

Our main result shows that such an ε-approximate gradient estimate (in the ℓ_∞-norm) can be computed with quadratically fewer *quantum* queries to a probability oracle. We remark that our quadratic speedup is not yet-another Grover speed-up, instead it comes from applying the Fourier transformation together with some optimized interpolation techniques.

[1] Note that this is a much weaker input model than the oracle model used by Jordan [28].

Our improved gradient computation algorithm. Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is given by an oracle, which on input $\boldsymbol{x}$, outputs $f(\boldsymbol{x})$ binarily with some finite accuracy. Jordan [28] constructed a quantum algorithm that outputs an ε-coordinate-wise approximation of the gradient ∇f using a single evaluation of the binary oracle. This demonstrates a striking quantum advantage, since classically one needs $\Omega(d)$ queries. His algorithm prepares a uniform superposition of evaluation points over a finite grid, then approximately implements a phase unitary

$$\mathrm{O}_f : |\boldsymbol{x}\rangle \mapsto e^{2\pi i \frac{\sqrt{d}}{\varepsilon^2} f(\boldsymbol{x})}|\boldsymbol{x}\rangle,$$

using a *single* $\mathcal{O}\big(\varepsilon^2/\sqrt{d}\big)$-accurate evaluation of f and then applies an inverse Fourier transformation to obtain an approximation of the gradient. Although this algorithm only uses a single query, the required precision of the function evaluation can be prohibitive. Moreover, the original analysis of Jordan [28] implicitly assumes that the function is essentially quadratic, by neglecting third and higher-order contributions. Using Jordan's algorithm in our framework, where we assume access to a probability oracle, evaluating the function with the prescribed precision using amplitude amplification would require $\Omega(\sqrt{d}/\varepsilon^2)$ queries.

Our improvements are two-fold: our quantum algorithm requires only $\widetilde{\mathcal{O}}(\varepsilon/\sqrt{d})$-accurate evaluations of the function; on the other hand it also works for functions with non-negligible higher-order terms, such as the objective functions arising from variational circuits. The main new ingredient is the use of higher-degree central-difference formulas, a technique borrowed from calculus. The heart of our proof is showing that if f is sufficiently smooth, then by using central-difference formulas we can approximately linearize the function on a diameter-1 hypercube by performing only $\log(\sqrt{d}/\varepsilon)$ function evaluations. We prove this by bounding the "second moment" of higher-order bounded tensors using Lemma 5.3, which might be of independent interest.

Our algorithm works by evaluating the approximately linearized function over a uniform superposition of grid points in the hypercube, followed by a d-dimensional quantum Fourier transform, providing a classical description of an approximate gradient, similarly to Jordan's algorithm.

THEOREM 1.1. (SEE THEOREM 5.4 & THEOREM 6.1) *Let $\varepsilon > 0$, $d \in \mathbb{N}$, and $c = \mathcal{O}(1)$. Suppose we are given probability (or phase) oracle access to a function $f : \mathbb{R}^d \to \mathbb{R}$, such that $|\partial_{i_1,i_2,\ldots,i_k} f(\mathbf{0})| \le c^k\sqrt{k!}$ for every $k \in \mathbb{N}$ and $(i_1, i_2, \ldots, i_k) \in [d]^k$. The quantum query complexity of computing (with high prob.) an ε-coordinatewise-approximation of $\nabla f(\mathbf{0})$ is*

$$\widetilde{\Theta}\Big(\sqrt{d}/\varepsilon\Big).$$

Our algorithm is also gate-efficient; the gate complexity is $\widetilde{\mathcal{O}}(Q+d)$, where Q is the query complexity.

Our lower bound techniques. In order to prove the optimality of our algorithm we prove an extended version of the so-called hybrid method [7].

THEOREM 1.2. (CONTINUOUS-PHASE LOWER BOUND) *Let $\mathcal{F}$ be a finite set of functions $X \to \mathbb{R}$. Suppose we have access to $f \in \mathcal{F}$ via a* phase oracle *such that*

$$\mathrm{O}_f : |x\rangle \mapsto e^{if(x)}|x\rangle \quad \text{for every } x \in X.$$

Let $f_* \in \mathcal{F}$ be fixed. Given O_f for some unknown $f \in \mathcal{F}$, determining whether $f = f_*$ has query complexity

$$\Omega\left(\sqrt{|\mathcal{F}|-1}\Big/\sqrt{\max_{x\in X}\sum_{f\in\mathcal{F}}|f(x)-f_*(x)|^2}\right).$$

In order to prove the lower bound in Theorem 1.1, we exhibit a family of functions $\mathcal{F}$ for which the functions can be well distinguished by calculating their gradients with accuracy ε in the ℓ_∞-norm. Then, with the help of Theorem 1.2 we show that this requires $\Omega(\sqrt{d}/\varepsilon)$ queries.

Applications. We consider three problems to which we apply our quantum gradient descent algorithm. We briefly describe below the problem of quantum variational eigensolvers (VQE) [39, 47], quantum approximate optimization algorithms (QAOA) [19], and the quantum auto-encoding problem [45, 41]. In each case we show how our gradient computation algorithm can provide a quadratic speedup in terms of the dimension d of the associated problem.

VQE is widely used to estimate the eigenvalue corresponding to some eigenstate of a Hamiltonian. The main idea in VQE is to begin with an efficiently parameterizable ansatz to the eigenstate. For the example of ground state energy estimation, the ansatz state is often taken to be a unitary coupled cluster expansion. The terms in that unitary coupled cluster expansion are varied to provide the lowest energy for the groundstate, and the expected energy of the quantum state is mapped to the probability of some measurement outcome, making it accessible to our methods.

QAOA has a similar approach, the basic idea of the algorithm is to consider a parametrized family of states such as $|\psi(\boldsymbol{x})\rangle = \prod_{j=1}^{d} e^{-ix_jH_j}|0\rangle$. The aim is to tune the parameters of $|\psi(\boldsymbol{x})\rangle$ in order to minimize some objective function, which can, e.g., represent some combinatorial optimization problem. In particular, if H is a Hermitian operator corresponding to the objective function then we wish to find $\boldsymbol{x}$ such that $\langle\psi(\boldsymbol{x})|H|\psi(\boldsymbol{x})\rangle$ is minimized. For example, in order to minimize the number of violated constraints of a constraint satisfaction problem, we can choose $H = \sum_{m=1}^{M} C_m$ to represent the number of violations: C_m is 1 if and only if the m^{th} constraint is violated, else $C_m = 0$ [19]. After normalization and using some standard techniques we can map this expectation value to some measurement probability. Thus, from the perspective of our algorithm, QAOA looks exactly like VQE.

The classical auto-encoder paradigm [5] is an important technique in machine learning, which is widely used for data compression. An auto-encoder is essentially a neural network architecture which is tuned for the following task: given a set of high-dimensional vectors, we would like to learn a low-dimensional representation of the vectors, so that computations on the original data set can be "approximately" carried out by working only with the low-dimensional representations. What makes auto-encoding powerful is that it does not assume any prior knowledge about the data set. This makes it a viable technique in machine learning, with various applications in natural language processing, training neural networks, object classification, prediction or extrapolation of information, etc. In this paper, we consider a natural quantum analogue (which was also considered before in the works of [45, 41]) of the auto-encoder paradigm, and show how to use our quantum gradient computation algorithm to quadratically speed up the training of quantum autoencoders.

2 Organization of the paper and preliminaries

In Section 3, we give a generic model of quantum optimization algorithms and a detailed description of the classical gradient descent algorithm. In Section 4, we describe how to convert a probability oracle to a phase oracle. In Section 5 we present our quantum gradient computation algorithm and prove our main Theorem 5.4 regarding its complexity. In Section 6, we present query lower bounds for algorithms that (approximately) compute the gradient of a function. In Section 7 we describe some applications. We conclude with some directions for future research in Section 8.

Notation. Let $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_d \in \mathbb{R}^d$ denote the standard basis vectors. We use bold letters for vectors $\boldsymbol{x} \in \mathbb{R}^d$, in particular we use the notation $\mathbf{0}$ for the 0 vector, and $\mathbf{1}$ for the all-1 vector $(\boldsymbol{e}_1 + \boldsymbol{e}_2 + \cdots + \boldsymbol{e}_d)$. By writing $\boldsymbol{y} + rS$ we mean $\{\boldsymbol{y} + r\boldsymbol{v} : \boldsymbol{v} \in S\}$ for vectors $S \subseteq \mathbb{R}^d$, and use the same notation for sets of numbers. For $\boldsymbol{x} \in \mathbb{R}^d$, let $\|\boldsymbol{x}\|_\infty = \max_{i\in[d]} |x_i|$ and $\|\boldsymbol{x}\| = (\sum_{i=1}^{d} x_i^2)^{1/2}$. For $M \in \mathbb{R}^{d\times d}$, let $\|M\|$ denote the operator norm of M.

Let $[d] = \{1, 2, \ldots, d\}$. We use the convention

$0^0 = 1$ throughout, and use the notation $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

When we state the complexity of an algorithm, we use the notation $\widetilde{\mathcal{O}}(C)$ to hide poly-logarithmic factors in the complexity C. In general, we use $\mathcal{H}$ to denote a finite dimensional Hilbert space. For the n-qubit all-0 basis state we use the notation $|0\rangle^{\otimes n}$, or simply write $|\vec{0}\rangle$ when we do not want to focus on the value of n.

Higher-order calculus. Many technical lemmas in this paper will revolve around the use of higher-order calculus. We briefly introduce some notation here and give some basic definitions.

DEFINITION 2.1. (INDEX-SEQUENCES) *For $k \in \mathbb{N}_0$ we call $\alpha \in [d]^k$ a d-dimensional length-k index-sequence. For a vector $\boldsymbol{r} \in \mathbb{R}^d$ we define $\boldsymbol{r}^\alpha := \prod_{j\in[k]} r_{\alpha_j}$. Also, for a k-times differentiable function, we define $\partial_\alpha f := \partial_{\alpha_1}\partial_{\alpha_2}\cdots\partial_{\alpha_k} f$. Finally, we use the notation $|\alpha| = k$ for denoting the length of the index-sequence.*

DEFINITION 2.2. (ANALYTIC FUNCTION) *We say that the function $f : \mathbb{R}^d \to \mathbb{R}$ is analytic if for all $\boldsymbol{x} \in \mathbb{R}^d$*

$$f(\boldsymbol{x}) = \sum_{k=0}^{\infty} \sum_{\alpha\in[d]^k} \boldsymbol{x}^\alpha \frac{\partial_\alpha f(\boldsymbol{0})}{k!}. \tag{2.4}$$

DEFINITION 2.3. (DIRECTIONAL DERIVATIVE) *Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is k-times differentiable at $\boldsymbol{x} \in \mathbb{R}^d$. We define the k-th order directional derivative in the direction $\boldsymbol{r} \in \mathbb{R}^d$ using the derivative of a one-parameter function parametrized by $\tau \in \mathbb{R}$ along the ray in the direction of $\boldsymbol{r}$:*

$$\partial_{\boldsymbol{r}}^k f(\boldsymbol{x}) = \frac{d^k}{(d\tau)^k} f(\boldsymbol{x} + \tau\boldsymbol{r}).$$

Observe that, using the definitions above, one has

$$\partial_{\boldsymbol{r}}^k f = \sum_{\alpha\in[d]^k} \boldsymbol{r}^\alpha \cdot \partial_\alpha f. \tag{2.5}$$

In particular for every $i \in [d]$, we have that $\partial_{\boldsymbol{e}_i}^k f = \partial_i^k f$.

Central difference formulas (see, e.g. [34]) are often used to give precise approximations of derivatives of a function $h : \mathbb{R} \to \mathbb{R}$. These formulas are coming from polynomial interpolation, and yield precise approximations of directional derivatives too. Thus, we can use them to approximate the gradient of a high-dimensional function as shown in the following definition.

DEFINITION 2.4. *The degree-$2m$* approximate central-difference linearization *of a function $f : \mathbb{R}^d \to \mathbb{R}$ is:*

$$f_{(2m)}(\boldsymbol{x}) := \sum_{\substack{\ell=-m \\ \ell\neq 0}}^{m} \frac{(-1)^{\ell-1}}{\ell} \frac{\binom{m}{|\ell|}}{\binom{m+|\ell|}{|\ell|}} f(\ell\boldsymbol{x}) \approx \nabla f(\boldsymbol{0}) \cdot \boldsymbol{x}. \tag{2.6}$$

We denote the corresponding central-difference coefficients *for $\ell \in \{-m, \ldots, m\}\backslash\{0\}$ by*

$$a_\ell^{(2m)} := \frac{(-1)^{\ell-1}}{\ell} \frac{\binom{m}{|\ell|}}{\binom{m+|\ell|}{|\ell|}} \quad \text{and } a_0^{(2m)} := 0$$

In the full version of this paper [22] we prove some bounds on the approximation error of the above formulas[2] for generic m. Usually such error bounds are only derived for some finite values of m, because that is sufficient in practice, but in order to prove our asymptotic results we need to derive more general results.

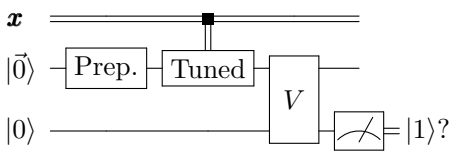
3 A generic model of quantum optimization algorithms

Variational quantum algorithms designed for quantum optimization and machine learning procedures have the following core idea: they approximate an optimal solution to a problem by tuning some parameters in a quantum circuit. The circuit usually consists of several simple gates, some of which have tunable real parameters, e.g., the angle of single qubit (controlled) rotation gates. Often, if there are enough tunable gates arranged in a nice topology, then there exists parameters that induce a unitary capable of achieving a close to optimal solution.

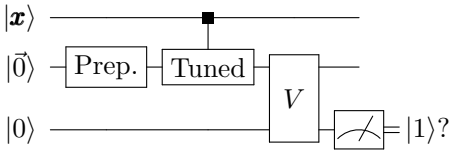
In such variational approaches, one can decompose the circuit into three parts each having a different role (see Figure 1). The circuit starts with a state preparation part which prepares the initial quantum state relevant for the problem. We call this part 'Prep.' in Figure 1. The middle part consists of tunable parameters x and fixed gates, which are together referred to as 'Tuned' in Figure 1. Finally, there is a verification circuit that evaluates the output state, and marks success if the auxiliary qubit is $|1\rangle$. We call the verification process V in Figure 1. The quality of the circuit (for parameter $\boldsymbol{x}$) is assessed by the probability of measuring the auxiliary qubit and obtaining 1.

One can think of the tunable circuit as being tuned in a classical way as shown in Figure 1a or a quantum way as in Figure 1b. In the classical case, the parameters can be thought of as being manually set. Alternatively, the parameters can be quantum variables represented by qubits. The advantage of the latter is that it allows us to use quantum techniques to speedup optimization algorithms. However the drawback is that it requires more qubits to represent the parameters and requires implementation of additional controlled-gates, see for example, Fig. 1c.

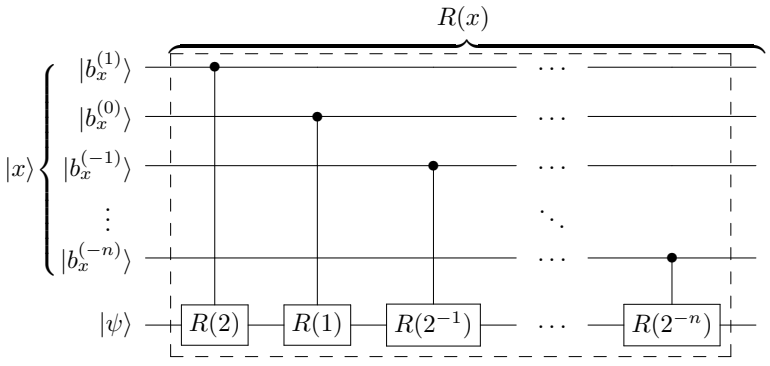
[2] One can read out the coefficients described in Definition 2.4 from the second row of the inverse of the Vandermonde matrix, as Arjan Cornelissen pointed out to us.



(a) A classically tunable circuit



(b) A quantumly tunable circuit



(c) A 2^{-n} precisely tunable rotation gate $R(x)$ for the fixed point binary parameter $x = b_1 b_0 . b_{-1} \cdots b_{-n}$.

Figure 1: Two different approaches to tunable quantum optimization. The circuit on the top left has classically set parameters $|\boldsymbol{x}\rangle$ (represented as a vector of fixed point binary numbers), whereas the circuit on the bottom left has parameters $\boldsymbol{x}$ described by an array of qubits $|\boldsymbol{x}\rangle$. The black squares connected to the 'Tuned' circuit indicate non-trivial control structure for which an example is presented on the right figure, showing how to implement a quantumly tunable rotation gate built from simple controlled rotation gates.

Let us denote by $U(\boldsymbol{x})$ the circuit in Figure 1a and the corresponding circuit in Figure 1b as $U := \sum_{\boldsymbol{x}} |\boldsymbol{x}\rangle\langle\boldsymbol{x}| \otimes U(\boldsymbol{x})$. The goal in these optimization problems is to find the optimal parameters (i.e., $\boldsymbol{x}$) which maximizes the probability of obtaining 1 after the final measurement, thereby solving the problem

$$\operatorname*{argmax}_{\boldsymbol{x}} p(\boldsymbol{x}), \text{ where } p(\boldsymbol{x}) = \left\| (|1\rangle\langle 1| \otimes I) U(\boldsymbol{x}) |\vec{0}\rangle \right\|^2. \tag{3.7}$$

A well-known technique to solve continuous-variable optimization problems like the one above is gradient-ascent. In practice, gradient-based methods represent one of the most commonly used paradigms for solving continuous optimization problems.

3.1 Classical gradient ascent algorithm As we discussed earlier, finding globally optimal parameters for optimization problems (3.7) is often hard. Therefore in practice one usually relies on heuristics to find an approximate solution $\boldsymbol{x}_a$ such that $p(\boldsymbol{x}_a)$ is close to optimal. There are several heuristic optimization techniques that are often applied to handle such problems. One of the most common techniques is gradient ascent, which follows a greedy strategy to obtain the optimal solution. It simply follows the path of steepest ascent on the landscape of the objective function to find a solution that is, at least up to small local perturbations, optimal. Such solutions are called locally optimal.

A naïve gradient-based algorithm[3] can be described as follows: pick N random points $\{\mathbf{x}_1^{(0)}, \ldots, \mathbf{x}_N^{(0)}\}$. For each $i \in [N]$, compute $\nabla p(\mathbf{x}_i^{(0)})$, and take a δ-step in the direction of $\nabla p(\mathbf{x}_i^{(0)})$ leading to $\mathbf{x}_i^{(1)} = \mathbf{x}_i^{(0)} + \delta \nabla p(\mathbf{x}_i^{(0)})$ (for some step size $\delta > 0$). Repeat this procedure for T steps, obtaining $\mathbf{x}_i^{(T)}$ which has hopefully approached some local maxima of (1.1). Finally, take the maximum of $\{p(\mathbf{x}_1^{(T)}), \ldots, p(\mathbf{x}_N^{(T)})\}$ as an approximation to (1.1).

By using empirical estimation to evaluate the function to precision ε, under some mild smoothness assumption, we can compute the gradient to ε-precision in the ℓ_∞ norm using $\widetilde{\mathcal{O}}\left(\frac{d}{\varepsilon^2}\right)$ samples. This way the algorithm uses the quantum circuit $U(\boldsymbol{x})$ a total number $\widetilde{\mathcal{O}}\left(NTd/\varepsilon^2\right)$ times.

3.2 Quantum speedups to the classical algorithm Now, let us consider the possible quantum speedups to this naïve gradient ascent algorithm discussed in the previous section. The most basic improvement which works even for classically controlled circuits (Figure 1a) is to estimate the probability $p(\boldsymbol{x})$ in Step 5 using quantum amplitude estimation rather than doing repeated measurements and taking the average. If one wants to determine the value $p(\boldsymbol{x})$ up to error ε for some fixed $\boldsymbol{x}$, the quantum approach uses the circuit $\mathcal{O}(1/\varepsilon)$ times, whereas the classical statistical method would require $\Omega(1/\varepsilon^2)$ repetitions, due to the additive property of variances of uncorrelated random variables. Although this is a natural improvement, which does not require much additional quantum resources many papers that describe a similar procedure do not mention it.

Another quantum improvement can be achieved [11, 33] by using Grover search, which requires a quantumly

[3] There are more sophisticated versions of gradient ascent, but for simplicity here we discuss a very basic version.

Method:	Simple algorithm	+Amp. est.	+Grover search	+**This paper**
Complexity:	$\widetilde{\mathcal{O}}(TNd/\varepsilon^2)$	$\widetilde{\mathcal{O}}(TNd/\varepsilon)$	$\widetilde{\mathcal{O}}\left(T\sqrt{N}d/\varepsilon\right)$	$\widetilde{\mathcal{O}}\left(T\sqrt{Nd}/\varepsilon\right)$

Table 1: Quantum speedups for the naïve gradient-ascent algorithm

controlled circuit like in Figure 1b. Let $P(\boldsymbol{z})$ denote the probability that for a randomly chosen starting point $\boldsymbol{x}_0$ we get $\boldsymbol{x}_T = \boldsymbol{z}$, i.e., we end up with $\boldsymbol{z}$ after performing T gradient steps. Let $\tilde{p}$ be a value such that $P(p(\boldsymbol{z}) \geq \tilde{p}) \geq 1/N$. If we use N randomly chosen initial points then with high probability at least one initial point will yield a point $\boldsymbol{x}_T$ with $p(\boldsymbol{x}_T) \geq \tilde{p}$.[4] If we use the quantum maximum finding algorithm [17] or more precisely one if its generalizations [37, 3], we can reduce the number of repetitions to $O(\sqrt{N})$ and still find a point $\boldsymbol{x}_T$ having $p(\boldsymbol{x}_T) \geq \tilde{p}$ with high probability. Due to reversability, we need to maintain all points visited during the gradient ascent algorithm, thereby possibly introducing a significant overhead in the number of qubits used.

However, there is a drawback using Grover search-based techniques. The disadvantage of quantum maximum finding approach over classical methods is, that the amount of time it takes to reach a local maximum using the gradient ascent might vary a lot. The reason is that classically, once we reached a local maximum we can start examining the next starting point, whereas if we use Grover search we do the gradient updates in superposition so we need to run the procedure for the largest possible number of gradient steps. To reduce this disadvantage one could use variable time amplitude amplification techniques introduced by Ambainis [1], however, we leave such investigations for future work.

Our contribution. We show a quadratic speedup in d – the number of control parameters. For this we also need to use a quantumly controlled circuit, but the overhead in the number of qubits is much smaller than in the previous Grover type speedup. The underlying quantum technique crucially relies on the quantum Fourier transform as it is based on an improved version of Jordan's gradient computation [28] algorithm. We can optionally combine this speedup with the above mentioned maximum finding improvement, which then gives a quantum algorithm that uses the quantumly controlled circuit (Figure 1b) $\widetilde{\mathcal{O}}\left(T\sqrt{Nd}/\varepsilon\right)$ times, and achieves essentially the same guarantees as the classical Algorithm. Therefore we can achieve a quadratic speedup in terms of all parameters except in T and obtain an overall complexity of $\widetilde{\mathcal{O}}\left(T\sqrt{Nd/\varepsilon}\right)$. For a summary of the speedups see Table 1.

[4] I.e., with high probability, we will find a point from the top $1/N$ percentile of the points regarding the objective function $p(\boldsymbol{z})$.

4 The three natural input oracle models

As we discussed in the previous section, the optimization problem associated with Figure 1 was to maximize

$$\max_{\boldsymbol{x}} p(\boldsymbol{x}) = \max_{\boldsymbol{x}} \left\| (|1\rangle\langle 1| \otimes I) U(\boldsymbol{x}) |\vec{0}\rangle \right\|^2.$$

Typically, one should think of U as the unitary corresponding to some variational quantum circuit or parametrized quantum algorithm. In more abstract terms, we can view U as a *probability oracle* that maps $|\vec{0}\rangle|\boldsymbol{x}\rangle$ to $\left(\sqrt{p(\boldsymbol{x})}|1\rangle|\psi_{\boldsymbol{x}}^{(1)}\rangle + \sqrt{1-p(\boldsymbol{x})}|0\rangle|\psi_{\boldsymbol{x}}^{(0)}\rangle\right)|\boldsymbol{x}\rangle$, such that the probability of obtaining 1 by measuring the first qubit is $p(x)$. This measurement probability serves as a "benchmark score" for the corresponding unitary U with respect to the vector of parameters $\boldsymbol{x}$.

DEFINITION 4.1. (PROBABILITY ORACLE) *We say that $U_p : \mathcal{H}_{aux.} \otimes \mathcal{H} \to \mathcal{H}_{aux.} \otimes \mathcal{H}$ is a probability oracle for the function $p : X \to [0,1]$, if $\{|x\rangle : x \in X\}$ is an orthonormal basis of the Hilbert space $\mathcal{H}$, and U_p for all $x \in X$ acts as*

$$|\vec{0}\rangle|x\rangle \mapsto \left(\sqrt{p(x)}|1\rangle|\psi_x^{(1)}\rangle + \sqrt{1-p(x)}|0\rangle|\psi_x^{(0)}\rangle\right)|x\rangle,$$

where $|\psi_x^{(1)}\rangle$ and $|\psi_x^{(0)}\rangle$ are arbitrary (normalized) quantum states.

This oracle model is not commonly used in quantum algorithms, therefore we need to convert it to a different format, for example to a *phase oracle*, in order to use standard quantum techniques.

DEFINITION 4.2. (PHASE ORACLE) *We say that $\mathrm{O}_f : \mathcal{H}_{aux.} \otimes \mathcal{H} \to \mathcal{H}_{aux.} \otimes \mathcal{H}$ is a phase oracle for $f : X \to [-1,1]$, if $\{|x\rangle : x \in X\}$ is an orthonormal basis of the Hilbert space $\mathcal{H}$, and O_f for all $x \in X$ acts as*

$$\mathrm{O}_f : |\vec{0}\rangle|x\rangle \mapsto e^{if(x)}|\vec{0}\rangle|x\rangle.$$

There is a third type of oracle that is commonly used in (quantum) algorithms, namely the *binary oracle* that outputs a finite precision binary representation of the function [48]. This is the input model used in the work of Jordan [28], but in fact the first step of his algorithm converts this oracle to a phase oracle.

DEFINITION 4.3. (BINARY ORACLE) *For $\eta \in \mathbb{R}_+$, we say $B_f^\eta : \mathcal{H} \otimes \mathcal{H}_{aux.} \to \mathcal{H} \otimes \mathcal{H}_{aux.}$ is an η-accurate binary oracle for $f : X \to \mathbb{R}$, if $\{|x\rangle : x \in X\}$ is an orthonormal basis of the Hilbert space $\mathcal{H}$, and for all $x \in X$ it acts as*

$$B_p^\eta : |x\rangle|\vec{0}\rangle \mapsto |x\rangle|p'(x)\rangle,$$

where $|p'(x)\rangle$ is a fixed-point binary number satisfying $|p'(x) - p(x)| \leq \eta$. We define the cost of one query to B_f^η as $C(\eta)$.[5,6]

Note that using amplitude estimation[7] it is possible to turn a probability oracle to a binary oracle and then convert a binary oracle to a phase oracle. However, it is more efficient to avoid this analogue-digital-analogue conversion, and directly convert a probability oracle to a phase oracle.

4.1 Converting a probability oracle to a (fractional) phase oracle We use block-encoding and Hamiltonian simulation techniques to implement the conversion efficiently. In order to describe our result we use the following definition, motivated by [23].

DEFINITION 4.4. (BLOCK-ENCODING) *Suppose that A is an operator on $\mathcal{H}$, then we say that the unitary U acting on $\mathcal{H}_{aux.} \otimes \mathcal{H}$ is a block-encoding of A, if*

$$A = (\langle\vec{0}| \otimes I)U(|\vec{0}\rangle \otimes I).$$

Intuitively a block-encoding is a unitary, whose top-left block contains the desired matrix A:

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix} \iff A = (\langle\vec{0}| \otimes I)U(|\vec{0}\rangle \otimes I).$$

Low and Chuang [35] showed how to implement an optimal Hamiltonian simulation circuit given a block-encoding of the Hamiltonian.

THEOREM 4.1. (HAMILTONIAN SIMULATION [35]) *Suppose that U is a block-encoding of the Hamiltonian H, and $\mathcal{H}_{aux.}$ consists of a qubits. Then one can implement an ε-precise approximation of the Hamiltonian simulation unitary e^{itH} using 2 additional ancilla qubits, with $\mathcal{O}(|t| + \log(1/\varepsilon))$ uses of controlled-U or its inverse and with $\mathcal{O}(a|t| + a\log(1/\varepsilon))$ two-qubit gates.*

We implement (fractional) phase oracles by first turning a probability oracle to a block-encoding of the diagonal matrix containing the probabilities, then using the above Hamiltonian simulation result.

Let U_p be a probability oracle, observe that

$$(\langle\vec{0}| \otimes I)\big(U_p^\dagger(Z \otimes I)U_p\big)(|\vec{0}\rangle \otimes I) = \mathrm{diag}(1 - 2p(x)).$$

Therefore $U_p^\dagger(Z \otimes I)U_p$ is a block-encoding of a diagonal matrix with diagonal entries $(1 - 2p(x))$. Moreover, we can implement a controlled a version of this unitary by simply replacing Z with a controlled-Z.

COROLLARY 4.1. (PROBABILITY TO PHASE ORACLE) *Suppose that we are given a probability oracle for $p(x) \colon X \to [0, 1]$. Let $t \in \mathbb{R}$ and $f(x) = tp(x)$ for all $x \in X$. We can implement an ε-approximate phase oracle O_f with query complexity $\mathcal{O}(|t| + \log(1/\varepsilon))$, i.e., this many uses of U_p and its inverse. Moreover, if $|\vec{0}\rangle = |0\rangle^{\otimes a}$ is an a-qubit state, then this query complexity bound times $\mathcal{O}(a)$ upper bounds the gate complexity.*

Form this we see that a probability oracle can be efficiently converted to a (fractional) phase oracle. Recently it was also shown, that if $|f| \leq 1$, then with a similar logarithmic overhead a phase oracle can be converted to fractional phase oracle [23]. For these reasons, and for simplicity, throughout the paper we assume that a fractional phase query has the same cost as a non-fractional/full phase query.

DEFINITION 4.5. (FRACTIONAL PHASE ORACLE) *Let $r \in [-1, 1]$, we say that $\mathrm{O}_{rf} : \mathcal{H}_{aux.} \otimes \mathcal{H} \to \mathcal{H}_{aux.} \otimes \mathcal{H}$ is a* fractional query[8] *phase oracle for $f : X \to [-1, 1]$, if $\{|x\rangle : x \in X\}$ is an orthonormal basis of the Hilbert space $\mathcal{H}$, and O_{rf} for all $x \in X$ acts as*

$$\mathrm{O}_{rf} : |\vec{0}\rangle|x\rangle \mapsto e^{irf(x)}|\vec{0}\rangle|x\rangle.$$

5 Improved quantum gradient computation algorithm

5.1 Overview of Jordan's algorithm Stephen Jordan constructed a surprisingly simple quantum algorithm [28, 12] that can approximately calculate the d-dimensional gradient of a function $f : \mathbb{R}^d \to \mathbb{R}$ with a *single* evaluation of f. In contrast, using standard

[5] One could also consider a more general definition allowing the oracle to output a superposition of η-accurate answers.

[6] The cost function would typically be polylog$(1/\eta)$ for functions that can be calculated using a classical circuit, however, when the binary oracle is obtained via quantum phase estimation this cost is typically $1/\eta$.

[7] In some cases people use sampling and classical statistics to learn this probability, however amplitude estimation is quadratically more efficient. Typically one can improve sampling procedures quadratically using quantum techniques [36, 25].

[8] Note that this fractional query is more general than the fractional query introduced by Cleve et al. [15], because we have a continuous phase rather than discrete. Thus, the results of [15] do not give a way to implement a generic fractional query using a simple phase oracle O_f, however one can use qubitization techniques in order to implement fractional queries [23].

classical techniques, one would use $d+1$ function evaluations to calculate the gradient at a point $\boldsymbol{x} \in \mathbb{R}^d$: one can first evaluate $f(\boldsymbol{x})$ and then, for every $i \in [d]$, evaluate $f(\boldsymbol{x}+\delta \boldsymbol{e}_i)$ (for some $\delta > 0$) to get an approximation of the gradient in direction i using the standard formula

$$\nabla_i f(\boldsymbol{x}) \approx \frac{f(\boldsymbol{x}+\delta \boldsymbol{e}_i) - f(\boldsymbol{x})}{\delta}.$$

The basic idea of Jordan's quantum algorithm [28] is simple. First make two observations. Observe that if f is twice differentiable at $\boldsymbol{x}$, then $f(\boldsymbol{x}+\boldsymbol{\delta}) = f(\boldsymbol{x}) + \nabla f \cdot \boldsymbol{\delta} + \mathcal{O}(\|\boldsymbol{\delta}\|^2)$, which in particular implies that for small $\|\boldsymbol{\delta}\|$, the function f is very close to being affine linear. The second observation is that, using the value of $f(\boldsymbol{x}+\boldsymbol{\delta})$, one can implement a phase oracle:

(5.8)
$$\mathrm{O}_{2\pi S f} : |\boldsymbol{\delta}\rangle \mapsto e^{2\pi i S f(\boldsymbol{x}+\boldsymbol{\delta})}|\boldsymbol{\delta}\rangle \approx e^{2\pi i S f(\boldsymbol{x})} e^{2\pi i S \nabla f \cdot \boldsymbol{\delta}}|\boldsymbol{\delta}\rangle$$

for a scaling factor $S > 0$, where the approximation uses $f(\boldsymbol{x}+\boldsymbol{\delta}) \approx f(\boldsymbol{x}) + \nabla f \cdot \boldsymbol{\delta}$ for small $\|\boldsymbol{\delta}\|$. The role of S is to make make the phases appropriate for the final quantum Fourier transform.

Sketch of the algorithm. Assume that all real vectors are expressed upto finite amount of precision. In order to compute the gradient at $\boldsymbol{x}$, the algorithm starts with a uniform superposition $|\psi\rangle = \frac{1}{\sqrt{|G_{\boldsymbol{x}}^d|}} \sum_{\boldsymbol{\delta} \in G_{\boldsymbol{x}}^d} |\boldsymbol{\delta}\rangle$ over the points of a sufficiently small discretized d-dimensional grid $G_{\boldsymbol{x}}^d$ around $\boldsymbol{x}$, and applies the phase oracle $\mathrm{O}_{2\pi S f}$ (in Eq. 5.8) to $|\psi\rangle$. Next, the inverse quantum Fourier transform is applied to the resulting state and each register is measured to obtain the gradient of f at $\boldsymbol{x}$ approximately. Due to approximate linearity of the phase (as in Eq. (5.8)), applying the Fourier transform will approximately give us the gradient. This algorithm uses $\mathrm{O}_{2\pi S f}$ once and Jordan showed how to implement $\mathrm{O}_{2\pi S f}$ using one sufficiently precise function evaluation.

In order to improve the accuracy of the simple algorithm above, one could use some natural tricks. If f is twice continuously differentiable, it is easy to see that the smaller the grid $G_{\boldsymbol{x}}^d$ becomes, the closer the function gets to being linear. This gives control over the precision of the algorithm, however if we "zoom-in" to the function using a smaller grid, the difference between nearby function values becomes smaller, making it harder the distinguish them and thus increasing the complexity of the algorithm proportionally.

Also, it is well known that if the derivative is calculated based on the differences between the points $(f(x-\delta/2),\ f(x+\delta/2))$ rather than $(f(x),\ f(x+\delta))$, then one gets a better approximation since the quadratic correction term cancels. To mimic this trick, Jordan chose a symmetric grid $G_{\boldsymbol{x}}^d$ around $\mathbf{0}$.

Complexity of the algorithm. For Jordan's algorithm, it remains to pick the parameters of the grid and the constant S in Eq. 5.8. For simplicity, assume that $\|\nabla f(\boldsymbol{x})\|_\infty \leq 1$, and suppose we want to approximate $\nabla f(\boldsymbol{x})$ coordinate-wise up to ε accuracy, with high success probability. Under the assumption that "the 2nd partial derivatives of f have a magnitude of approximately D_2", Jordan argues[9] that choosing $G_{\boldsymbol{x}}^d$ to be a d-dimensional hypercube with edge length $\ell \approx \frac{\varepsilon}{D_2\sqrt{d}}$ and with $N \approx \frac{1}{\varepsilon}$ equally spaced grid points in each dimension, the quantum algorithm yields an ε-approximate gradient by setting $S = \frac{N}{\ell} \approx \frac{D_2\sqrt{d}}{\varepsilon^2}$. Moreover, since the Fourier transform is relatively insensitive to local phase errors it is sufficient to implement the phase $Sf(\boldsymbol{x}+\boldsymbol{\delta})$ upto some constant, say 1% accuracy.

During the derivation of the above parameters Jordan makes the assumption, that the third and higher-order terms of the Taylor expansion of f around $\boldsymbol{x}$ are negligible, however it is not clear from his work [28], how to actually handle the case when they are non-negligible. This could be a cause of concern for the runtime analysis, since these higher-order terms potentially introduce a dependence on the dimension d.

Finally, in order to assess the complexity of his algorithm, Jordan considers the Binary oracle input model of Definiton 4.3. This input model captures functions that are evaluated numerically using, say, an arithmetic circuit. Typically, the number of one and two-qubit gates needed to evaluate such functions up to n digits precision is polynomial in n and d. However, this input model does not fit the quantum optimization framework that we introduced in Section 3.

Our improvements. We improve on the results of Jordan [28] in a number of ways. Jordan [28] argued that evaluating the function on a superposition of grid-points symmetrically arranged around $\mathbf{0}$ is analogous to using a simple central-difference formula. We also place the grid symmetrically, but we realized that it is possible to directly use central-difference formulas, which is the main idea behind our modified algorithm.

We realized that in applications of the gradient descent algorithm for optimization, it is natural to assume access to a phase oracle $\mathrm{O}_f : |\boldsymbol{x}\rangle \mapsto e^{if(\boldsymbol{x})}|\boldsymbol{x}\rangle$ (allowing fractional queries as well – see Definition 4.5) instead of the binary access oracle B_f^η. If we wanted to use Jordan's original algorithm in order to obtain the gradient with accuracy ε, we need to implement the

[9] We specifically refer to equation (4) in [28] (equation (3) in the arXiv version), and the discussion afterwards. Note that our precision parameter ε corresponds to the uncertainty parameter σ in [28].

query oracle O_f^S by setting $S \approx D_2\sqrt{d}/\varepsilon^2$, which can be achieved using $\lceil S \rceil$ consecutive (fractional) queries. Although it gives a square-root dependence on d, it scales as $\mathcal{O}(1/\varepsilon^2)$ with the precision. Here, we employ the phase oracle model and improve the quadratic dependence on $1/\varepsilon$ to essentially linear. Additionally, we rigorously prove the square-root scaling with d under reasonable assumptions on the derivatives of f. We also show that, for a class of smooth functions, the $\widetilde{\mathcal{O}}\left(\sqrt{d}/\varepsilon\right)$-query complexity is optimal up to polylogarithmic factors. We describe the algorithm in the next section, but first present our main result, whose proof is deferred to the end of this section.

5.2 Analysis of Jordan's algorithm In this section we describe Jordan's algorithm and provide a generic analysis of its behaviour. In the next subsection we combine these results with our finite-difference methods. Before describing the algorithm, we introduce appropriate representation of our qubit strings suitable for fixed-point arithmetics.

DEFINITION 5.1. *For every $b \in \{0,1\}^n$, let $j^{(b)} \in \{0,\dots,2^n-1\}$ be the integer corresponding to the binary string $b = (b_1,\dots,b_n)$. We label the n-qubit basis state $|b_1\rangle|b_2\rangle\cdots|b_n\rangle$ by $|x^{(b)}\rangle$, where*

$$x^{(b)} = \frac{j^{(b)}}{2^n} - \frac{1}{2} + 2^{-n-1}.$$

We denote the set of corresponding labels as $G_n := \left\{\frac{j^{(b)}}{2^n} - \frac{1}{2} + 2^{-n-1} : j^{(b)} \in \{0,\dots,2^n-1\}\right\}$. Note that there is a bijection between $\{j^{(b)}\}_{b\in\{0,1\}^n}$ and $\{x^{(b)}\}_{b\in\{0,1\}^n}$, so we use $|x^{(b)}\rangle$ and $|j^{(b)}\rangle$ interchangeably in Remark 5.1. In the rest of this section we always label n-qubit basis states by elements of G_n.

DEFINITION 5.2. *For $x \in G_n$ we define the Fourier transform of a state $|x\rangle$ as*

$$QFT_{G_n} : |x\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k\in G_n} e^{2\pi i 2^n xk}|k\rangle.$$

We prove the following in the full version of this paper [22]:

CLAIM 5.1. *This unitary is the same as the usual quantum Fourier transform up to conjugation with a tensor product of n single-qubit unitaries.*

Now we are ready to precisely describe Jordan's quantum gradient computation algorithm.

Algorithm 1 Jordan's quantum gradient computation algorithm

Registers: Use n-qubit input registers $|x_1\rangle|x_2\rangle\cdots|x_d\rangle$ with each qubit set to $|0\rangle$.
Labels: Label the n-qubit states of each register with elements of G_n as in Definition 5.1.
Input: A function $f : G_n^d \to \mathbb{R}$ with phase-oracle O_f access such that

$$\mathrm{O}_f^{\pi 2^{n+1}}|x_1\rangle\cdots|x_d\rangle = e^{2\pi i 2^n f(x_1,x_2,\dots,x_d)}|x_1\rangle\cdots|x_d\rangle.$$

1: **Init** Apply a Hadamard transform to each qubit of the input registers.
2: **Oracle call** Apply the modified phase oracle $\mathrm{O}_f^{\pi 2^{n+1}}$ on the input registers.
3: **QFT**$_{G_n}^{-1}$ Fourier transform each register:

$$|x\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k\in G_n} e^{-2\pi i 2^n xk}|k\rangle.$$

4: **Measure** each input register j and denote the measurement outcome by k_j.
5: **Output** $(k_1, k_2, \dots, k_d)$ as the gradient estimate

LEMMA 5.1. *Let $N = 2^n$, $c \in \mathbb{R}$ and $\boldsymbol{g} \in \mathbb{R}^d$ such that $\|\boldsymbol{g}\|_\infty \le 1/3$. If $f : G_n^d \to \mathbb{R}$ is such that*

$$|f(\boldsymbol{x}) - \boldsymbol{g}\cdot\boldsymbol{x} - c| \le \frac{1}{42\pi N}, \tag{5.9}$$

for all but a $1/1000$ fraction of the points $\boldsymbol{x} \in G_n^d$, then the output of Algorithm 1 satisfies:

$$\Pr[|k_i - g_i| > 4/N] \le 1/3 \quad \text{for every } i \in [d].$$

Proof. First, note that $|G_n| = N$ from Definition 5.1. Consider the following quantum states

$$|\phi\rangle = \frac{1}{\sqrt{N^d}} \sum_{\boldsymbol{x}\in G_n^d} e^{2\pi i N f(\boldsymbol{x})}|\boldsymbol{x}\rangle,$$
$$|\psi\rangle = \frac{1}{\sqrt{N^d}} \sum_{\boldsymbol{x}\in G_n^d} e^{2\pi i N(\boldsymbol{g}\cdot\boldsymbol{x}+c)}|\boldsymbol{x}\rangle.$$

Note that $|\phi\rangle$ is the state we obtain in Algorithm 1 after line 2 and $|\psi\rangle$ is its "ideal version" that we try to approximate with $|\phi\rangle$. Observe that the "ideal" $|\psi\rangle$ is actually a product state:

$$|\psi\rangle = \bigotimes_{\ell=1}^{d} \left(\frac{1}{\sqrt{N}} \sum_{x_\ell\in G_n} e^{2\pi i N g_\ell \cdot x_\ell}|x_\ell\rangle\right)$$

It is easy to see that after applying the inverse Fourier transform to each register in $|\psi\rangle$, we obtain

$$\bigotimes_{\ell=1}^{d}\Big(\frac{1}{N}\sum_{x_\ell,k_\ell\in G_n^2} e^{2\pi i N x_\ell(g_\ell-k_\ell)}|k_\ell\rangle\Big)$$

Suppose we make a measurement and observe $(k_1,\dots,k_d)$. As shown in the analysis of phase estimation [38], we have[10]: for every $i\in[d]$ (for a fixed accuracy parameter $\kappa>1$), the following holds:

$$\Pr\Big[|k_i-g_i|>\frac{\kappa}{N}\Big]\le\frac{1}{2(\kappa-1)}\qquad\text{for every } i\in[d].$$

By fixing $\kappa=4$, we obtain the desired conclusion of the theorem, i.e., if we had access to $|\psi\rangle$ (instead of $|\phi\rangle$), then we would get a $4/N$-approximation of each coordinate of the gradient with probability at least $5/6$. It remains to show that this probability does not change more than $1/3-1/6=1/6$ if we apply the Fourier transform to $|\phi\rangle$ instead of $|\psi\rangle$. Observe that the difference in the probability of any measurement outcome on these states is bounded by twice the trace distance between $|\psi\rangle$ and $|\phi\rangle$ which is

$$\||\psi\rangle\langle\psi|-|\phi\rangle\langle\phi|\|_1=2\sqrt{1-|\langle\psi|\phi\rangle|^2}\le 2\||\psi\rangle-|\phi\rangle\|.$$

Since the Fourier transform is unitary and does not change the Euclidean distance, it is sufficient to show that $\||\psi\rangle-|\phi\rangle\|\le 1/12$ in order to conclude the theorem. Let $S\subseteq G_n^d$ denote the set of points satisfying Eq. (5.9). We conclude the proof of the theorem by showing $\||\psi\rangle-|\phi\rangle\|^2\le(1/12)^2$: we upper bound $\||\phi\rangle-|\psi\rangle\|^2$ as follows

$$\begin{aligned}
&\frac{1}{N^d}\sum_{\boldsymbol{x}\in G_n^d}\Big|e^{2\pi iNf(\boldsymbol{x})}-e^{2\pi iN(\boldsymbol{g}\cdot\boldsymbol{x}+c)}\Big|^2\\
&=\frac{1}{N^d}\sum_{\boldsymbol{x}\in S}\Big|e^{2\pi iNf(\boldsymbol{x})}-e^{2\pi iN(\boldsymbol{g}\cdot\boldsymbol{x}+c)}\Big|^2\\
&\quad+\frac{1}{N^d}\sum_{\boldsymbol{x}\in G_n^d\setminus S}\Big|e^{2\pi iNf(\boldsymbol{x})}-e^{2\pi iN(\boldsymbol{g}\cdot\boldsymbol{x}+c)}\Big|^2\\
&\le\frac{1}{N^d}\sum_{\boldsymbol{x}\in S}|2\pi Nf(\boldsymbol{x})-2\pi N(\boldsymbol{g}\cdot\boldsymbol{x}+c)|^2+\frac{1}{N^d}\sum_{\boldsymbol{x}\in G_n^d\setminus S}4\\
&=\frac{1}{N^d}\sum_{\boldsymbol{x}\in S}(2\pi N)^2|f(\boldsymbol{x})-(\boldsymbol{g}\cdot\boldsymbol{x}+c)|^2+4\frac{|G_n^d\setminus S|}{N^d}\\
&\le\frac{1}{N^d}\sum_{\boldsymbol{x}\in S}\Big(\frac{1}{21}\Big)^2+\frac{4}{1000}\le\frac{1}{441}+\frac{1}{250}<\Big(\frac{1}{12}\Big)^2,
\end{aligned}$$

where the first inequality used $|e^{iz}-e^{iy}|\le|z-y|$, and the second inequality is by the assumptions on f. □

In the following theorem we assume that we have access to (a high power of) a phase oracle of a function f that is very well approximated by an affine linear function $\boldsymbol{g}\cdot\boldsymbol{z}+c$ on a hypergrid with edge-length $r\in\mathbb{R}$ around some $\boldsymbol{y}\in\mathbb{R}^d$. We show that if the relative precision of the approximation is precise enough, then Algorithm 1 can compute an approximation of $\boldsymbol{g}$ (the "gradient") with small query and gate complexity.

THEOREM 5.1. *Let $c\in\mathbb{R}$, $r,\rho,\varepsilon<M\in\mathbb{R}_+$, and $\boldsymbol{y},\boldsymbol{g}\in\mathbb{R}^d$ such that $\|g\|_\infty\le M$. Let $n_\varepsilon:=\lceil\log_2(4/(r\varepsilon))\rceil$, $n_M:=\lceil\log_2(3rM)\rceil$ and $n:=n_\varepsilon+n_M$. Suppose $f:(\boldsymbol{y}+rG_n^d)\to\mathbb{R}$ is such that*

$$|f(\boldsymbol{y}+r\boldsymbol{x})-\boldsymbol{g}\cdot r\boldsymbol{x}-c|\le\frac{\varepsilon r}{8\cdot 42\pi}$$

for all but a $1/1000$ fraction of the points $\boldsymbol{x}\in G_n^d$. If we have access to a phase oracle $\mathrm{O}:|\boldsymbol{x}\rangle\to e^{2\pi i2^{n_\varepsilon}f(\boldsymbol{y}+r\boldsymbol{x})}|\boldsymbol{x}\rangle$ acting on $\mathcal{H}=\mathrm{Span}\{|\boldsymbol{x}\rangle:\boldsymbol{x}\in G_n^d\}$, then using Algorithm 1 we can calculate a vector $\tilde{\boldsymbol{g}}\in\mathbb{R}^d$ such that

$$\Pr[\,\|\tilde{\boldsymbol{g}}-\boldsymbol{g}\|_\infty>\varepsilon]\le\rho,$$

with $\mathcal{O}\Big(\log\big(\frac{d}{\rho}\big)\Big)$ queries to O and with gate complexity $\mathcal{O}\Big(d\log\big(\frac{d}{\rho}\big)\log\big(\frac{M}{\varepsilon}\big)\log\log\big(\frac{d}{\rho}\big)\log\log\big(\frac{M}{\varepsilon}\big)\Big)$.

Proof. Let $N_M:=2^{n_M}$, $N:=2^n$, and $h(\boldsymbol{x}):=\frac{f(\boldsymbol{y}+r\boldsymbol{x})}{N_M}$, then $\big|h(x)-\boldsymbol{g}\frac{r}{N_M}\boldsymbol{x}-\frac{c}{N_M}\big|\le\frac{\varepsilon r}{8\cdot42\pi N_M}\le\frac{1}{42\pi N}$. Note that $\mathrm{O}=\mathrm{O}_h^{2\pi N}$, therefore Algorithm 1 yields and output $\tilde{\boldsymbol{g}}$, which, as shown by Lemma 5.1, is such that that for each $i\in[d]$ with probability at least $2/3$ we have $\big|\tilde{g}_i-\frac{r}{N_M}g_i\big|\le\frac{4}{N}$. Thus also $\big|\frac{N_M}{r}\tilde{g}_i-g_i\big|\le\frac{4N_M}{rN}\le\varepsilon$. By repeating the procedure $\mathcal{O}(\log(d/\rho))$ times and taking the median coordinate-wise we get a vector $\tilde{\boldsymbol{g}}_{\mathrm{med}}$, such that $\|\tilde{\boldsymbol{g}}_{\mathrm{med}}-\boldsymbol{g}\|_\infty\le\varepsilon$ with probability at least $(1-\rho)$.

The gate complexity statement follows from the fact that the complexity of Algorithm 1 is dominated by that of the d independent quantum Fourier transforms, each of which can be approximately implemented using $\mathcal{O}(n\log n)$ gates. We repeat the procedure $\mathcal{O}(\log(d/\rho))$ times, which amounts to $\mathcal{O}(d\log(d/\rho)n\log n)$ gates. At the end we get d groups of numbers each containing $\mathcal{O}(\log(d/\rho))$ numbers with n bits of precision. We can sort each group with a circuit having $\mathcal{O}(\log(d/\rho)\log\log(d/\rho)n\log n)$ gates.[11] So the final gate complexity is $\mathcal{O}(d\log(d/\rho)\log\log(d/\rho)n\log n)$, which gives the stated gate complexity by observing that $n=\mathcal{O}(\log(M/\varepsilon))$. □

[10] Note that our Fourier transform is slightly altered, but the same proof applies as in [38, (5.34)]. In fact this result can be directly translated to our case by considering the unitary conjugations proven in Remark 5.1.

[11] Note that using the median of medians algorithm [8] we could do this step with $\mathcal{O}(\log(d/\rho)n)$ time complexity, but this result probably does not apply to the circuit model, which is somewhat weaker than e.g. a Turing machine.

5.3 Improved quantum gradient algorithm using higher-degree methods As Theorem 5.1 shows Jordan's algorithm works well if the function is very close to linear function over a large hyprecube. However, in general even highly regular functions tend to quickly diverge from their linear approximations. To tackle this problem we borrow ideas from numerical analysis and use higher-degree finite-difference formulas to extend the range of approximate linearity.

We will apply Jordan's algorithm to the approximate finite-difference linearization of the function rather than the function itself. We illustrate the main idea on a simple example: suppose we want to calculate the gradient at $\mathbf{0}$, then we could use the 2-point approximation $(f(\boldsymbol{x}) - f(-\boldsymbol{x}))/2$ instead of f, which has the advantage that it cancels out even order contributions. The corresponding phase oracle $|\boldsymbol{x}\rangle \to e^{2^n \pi i (f(\boldsymbol{x}) - f(-\boldsymbol{x}))}|\boldsymbol{x}\rangle$ is also easy to implement as the product of the oracles:

$$+ \text{ phase oracle: } |\boldsymbol{x}\rangle = e^{2^n \pi i f(\boldsymbol{x})}|\boldsymbol{x}\rangle \quad \text{and}$$
$$- \text{ phase oracle: } |\boldsymbol{x}\rangle = e^{-2^n \pi i f(-\boldsymbol{x})}|\boldsymbol{x}\rangle.$$

We use high-order central-difference approximation formulas. There are a variety of other related formulas [34], but we stick to the central difference because the absolute values of the coefficients in this formula scale favorably with the approximation degree. Since we only consider central differences, all our approximations have even degree, which is sufficient for our purposes as we are interested in the asymptotic scaling. Nevertheless, it is not difficult to generalize our approach using other formulas [34] that can provide odd-degree approximations as well.

Algorithm 2 Improved quantum gradient computation

Input: A point $\boldsymbol{x} \in \mathbb{R}^d$ and a function $f : \mathbb{R}^d \to \mathbb{R}$ with probability or (fractional) phase oracle access
Parameters: m : interpolation order; R : rescaling
Function transformation: $g(\boldsymbol{y}) := R \cdot f(\boldsymbol{x} + \boldsymbol{y}/R)$
Oracle implementation:

$$\mathrm{O}^t_{g_{(2m)}} := \prod_{\ell=-m}^{m} \Big(I \otimes S_\ell^\dagger\Big) \cdot O_f^{tRa_\ell^{(2m)}} \cdot \Big(I \otimes S_\ell\Big),$$

where $O_f^{tRa_\ell^{(2m)}}$ is implemented by Corollary 4.1 or as a product of (fractional) phase oracles; I acts on the ancilla qubits used by $O_f^{tRa_\ell^{(2m)}}$, and $S_\ell \colon |\boldsymbol{y}\rangle \mapsto |\boldsymbol{x} + \ell\boldsymbol{y}/R\rangle$ for all $\boldsymbol{y} \in G$ evaluation points used by Algorithm 1
Use Jordan's Algorithm 1: compute $\nabla g_{(2m)}(\mathbf{0})$
Output: Approximation of $\nabla g_{(2m)}(\mathbf{0}) = \nabla f(\boldsymbol{x})$

In the full version of this paper [22] we prove the following lemma, showing that if $f : \mathbb{R} \to \mathbb{R}$ is $(2m+1)$-times continuously differentiable, then the central-difference formula in Eq. (2.6) is a good approximation to $f'(0)$. Eventually we also generalize this to the setting where $f : \mathbb{R}^d \to \mathbb{R}$ is higher-dimensional.

LEMMA 5.2. *Let $\delta \in \mathbb{R}_+$, $m \in \mathbb{N}$ and suppose $f : [-m\delta, m\delta] \to \mathbb{R}$ is $(2m+1)$-times differentiable. Then*

(5.10)
$$\begin{aligned} \big|f'(0)\delta - f_{(2m)}(\delta)\big| &= \left|f'(0)\delta - \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell\delta)\right| \\ &\le e^{-\frac{m}{2}} \left\|f^{(2m+1)}\right\|_\infty |\delta|^{2m+1}, \end{aligned}$$

where $\left\|f^{(2m+1)}\right\|_\infty := \sup_{\xi \in [-m\delta, m\delta]} |f^{(2m+1)}(\xi)|$ and $a_\ell^{(2m)}$ is defined in Definition 2.4. Moreover

$$\sum_{\ell=0}^{m} \left|a_\ell^{(2m)}\right| < \sum_{\ell=1}^{m} \frac{1}{\ell} \le \ln(m) + 1. \tag{5.11}$$

This lemma shows that for small enough δ the approximation error in (2.6) is upper bounded by a factor proportional to δ^{2m+1}. If $\left\|f^{(2m+1)}\right\|_\infty \le c^m$ for all m and we choose $\delta \le 1/c$, then the approximation error becomes exponentially small in m, motivating the use of higher-degree methods in our modified gradient computation algorithm. We generalize this statement to higher dimensions in the full version of this paper [22], which leads to our first result regarding our improved algorithm: (for the definition of the directional partial derivative $\partial_{\boldsymbol{r}}^{2m+1} f$ see Definition 2.3)

THEOREM 5.2. *Let $m \in \mathbb{Z}_+$, $D \in \mathbb{R}_+$ and $B \ge 0$. Suppose $f : [-D, D]^d \to \mathbb{R}$ is given with phase oracle access. If f is $(2m+1)$-times differentiable and for all $\boldsymbol{x} \in [-D, D]^d$ we have that*

$$|\partial_{\boldsymbol{r}}^{2m+1} f(\boldsymbol{x})| \le B \quad \text{for } \boldsymbol{r} = \boldsymbol{x}/\|\boldsymbol{x}\|,$$

then using Algorithm 2 with setting

$$R = \Theta\left(\max\left(\sqrt{d}\sqrt[2m]{\frac{B\sqrt{d}}{\varepsilon}}, \frac{m}{D}\right)\right)$$

we can compute an approximate gradient $\boldsymbol{g}$ such that $\|\boldsymbol{g} - \nabla f(\mathbf{0})\|_\infty \le \varepsilon$ with probability at least $(1-\rho)$, using $\mathcal{O}\Big(\big(\frac{R}{\varepsilon}\log(2m) + m\big)\log\Big(\frac{d}{\rho}\Big)\Big)$ phase queries.

Suppose that $D = \Theta(1)$, and f is a multi-variate polynomial of degree k. Then for $m = \lceil k/2 \rceil$ we get that $B = 0$, as can be seen by using (2.5), therefore the

above result gives an $\widetilde{\mathcal{O}}\left(\frac{k}{\varepsilon}\log\left(\frac{d}{\rho}\right)\right)$ query algorithm. If $2 \leq k = \mathcal{O}(\log(d))$, then this result gives an exponential speedup in terms of the dimension d compared to Jordan's algorithm. For comparison note that other recent results concerning quantum gradient descent also work under the assumption that the objective function is a polynomial [40, 30].

However, we argue in the full version of this paper [22] that for non-polynomial functions we can have $B \approx d^m$ even under strong regularity conditions. This then results in an $\widetilde{\mathcal{O}}\left(\frac{d}{\varepsilon}\right)$ query algorithm, achieving the desired scaling in ε but failing to capture the sought $\sqrt{d}$ scaling. In order to tackle the non-polynomial case we need to introduce some smoothness conditions.

5.4 Smoothness conditions and approximation error bounds In this subsection we show how to improve the result of Theorem 5.2, assuming some smoothness conditions. The calculus gets a bit involved, because we need to handle higher-dimensional analysis. In order to focus on the main results, we state the main result here and refer an interested reader to the full version of this paper [22]. We show that under reasonable smoothness assumptions, the complexity of our quantum algorithm is $\widetilde{O}(\sqrt{d}/\varepsilon)$ and in the next section show that for a specific class of smooth functions this is in fact optimal up to polylogarithmic factors.

A key ingredient of our proof is the following lemma, bounding the "second moment" of higher-order bounded tensors, proven in the full version of this paper [22].

LEMMA 5.3. *Let $d, k \in \mathbb{N}_+$, and suppose $H \in \left(\mathbb{R}^d\right)^{\otimes k}$ is an order k tensor of dimension d, having all elements bounded by 1 in absolute value, i.e., $\|H\|_\infty \leq 1$. Suppose $\{x_1, \ldots, x_d\}$ are i.i.d. symmetric random variable bounded in $[-1/2, 1/2]$ and satisfying $\mathbb{E}[(x_i)^{2k-1}] = 0$ for every $k \in \mathbb{N}_+$. Then $\mathbb{P}\left[\left|\sum_{\alpha\in[d]^k} H_\alpha x^\alpha\right| \geq \sqrt{2}\left(r\sqrt{\frac{dk}{2}}\right)^k\right] \leq \frac{1}{r^{2k}}$.*

Using this lemma, in the full version of the paper [22] we prove the following result about analytic functions:[12]

THEOREM 5.3. *If $r \in \mathbb{R}_+$, $f : \mathbb{R}^d \to \mathbb{R}$ is analytic and for all $k \in \mathbb{N}, \alpha \in [d]^k$ we have*

$$|\partial_\alpha f(\mathbf{0})| \leq c^k k^{\frac{k}{2}},$$

then

$$|\nabla f(\mathbf{0})\boldsymbol{y} - f_{(2m)}(\boldsymbol{y})| \leq \sum_{k=2m+1}^{\infty}\left(8rcm\sqrt{d}\right)^k,$$

for all but a 1/1000 fraction of points $\boldsymbol{y} \in r \cdot G_n^d$.

We can use this result to analyze the complexity of Algorithm 1 when applied to functions evaluated with using a central-difference formula. In particular it makes it easy to prove the following theorem, which is one of our main results.

THEOREM 5.4. *Let $\boldsymbol{x} \in \mathbb{R}^d$, $\varepsilon \leq c \in \mathbb{R}_+$ be fixed constants and suppose $f : \mathbb{R}^d \to \mathbb{R}$ is analytic[13] and satisfies the following: for every $k \in \mathbb{N}$ and $\alpha \in [d]^k$*

$$|\partial_\alpha f(\boldsymbol{x})| \leq c^k k^{\frac{k}{2}}.$$

Using Algorithm 2 with setting $m = \log(c\sqrt{d}/\varepsilon)$ and

$$R = \Theta\left(cm\sqrt{d}\right)$$

we can compute an ε-approximate gradient $\tilde{\nabla} f(\boldsymbol{x}) \in \mathbb{R}^d$ such that

$$\left\|\nabla f(\boldsymbol{x}) - \tilde{\nabla} f(\boldsymbol{x})\right\|_\infty \leq \varepsilon,$$

with probability at least $1 - \delta$, using $\widetilde{\mathcal{O}}\left(\frac{c\sqrt{d}}{\varepsilon}\log\left(\frac{d}{\delta}\right)\right)$ queries to a probability or (fractional) phase oracle of f.

Proof. Let $g(\boldsymbol{y}) := f(\boldsymbol{x} + \boldsymbol{y})$. By Theorem 5.3 we know that for a uniformly random $\boldsymbol{y} \in r \cdot G_n^d$ we have $|\nabla g(\mathbf{0})\boldsymbol{y} - g_{(2m)}(\boldsymbol{y})| \leq \sum_{k=2m+1}^{\infty}\left(8rcm\sqrt{d}\right)^k$ with probability at least 999/1000. Now we choose r such that this becomes smaller that $\frac{\varepsilon r}{8\cdot 42\pi}$. Now let us define $R := r^{-1} := 9cm\sqrt{d}\left(81\cdot 8\cdot 42\pi cm\sqrt{d}/\varepsilon\right)^{1/(2m)}$, then we get $8rcm\sqrt{d} = \frac{8}{9}\left(81\cdot 8\cdot 42\pi cm\sqrt{d}/\varepsilon\right)^{-1/(2m)}$ and so

$$\begin{aligned}
&\sum_{k=2m+1}^{\infty}\left(8rcm\sqrt{d}\right)^k \\
&= \left(8rcm\sqrt{d}\right)^{2m+1}\sum_{k=0}^{\infty}\left(8rcm\sqrt{d}\right)^k \\
&\leq \frac{\varepsilon}{81\cdot 8\cdot 42\pi cm\sqrt{d}}\left(81\cdot 8\cdot 42\pi cm\sqrt{d}/\varepsilon\right)^{\frac{-1}{2m}}\sum_{k=0}^{\infty}\left(\frac{8}{9}\right)^k \\
&= \frac{\varepsilon}{9cm\sqrt{d}\cdot 8\cdot 42\pi}\left(81\cdot 8\cdot 42\pi cm\sqrt{d}/\varepsilon\right)^{\frac{-1}{2m}} = \frac{\varepsilon r}{8\cdot 42\pi},
\end{aligned}$$

[12] The functions examined in the following two theorems are essentially Gevrey class $G^{\frac{1}{2}}$ functions [21].

[13] For convenience we assume in the statement that f can be evaluated at any point of $\mathbb{R}^d$, but in fact we only evaluate it inside a finite ball around $\boldsymbol{x}$. It is straightforward to translate the result when the function is only accessible on an open ball around $\boldsymbol{x}$. However, a finite domain imposes restrictions to the evaluation points of the function. If $\boldsymbol{x}$ lies too close to the boundary, this might impose additional scaling requirements and thus potentially increases the complexity of the derived algorithm. Fortunately in our applications it is natural to assume that f can be evaluated at distant points too, so we don't need worry about this detail.

where the inequality is by the choice of r, and the following equality uses $\sum_{k=0}^{\infty}\left(\frac{8}{9}\right)^k = 9$. By Theorem 5.1 we can compute an ε-approximation of $\nabla g_{(2m)}(\boldsymbol{x})$ with $\mathcal{O}(\log(d/\delta))$ queries to $\mathrm{O}^S_{g_{(2m)}}$, where $S = \mathcal{O}\left(\frac{1}{\varepsilon r}\right)$. Observe that

$$\mathrm{O}^S_{g_{(2m)}}|\boldsymbol{y}\rangle = e^{iSg_{(2m)}(\boldsymbol{y})}|\boldsymbol{y}\rangle = e^{iS\sum_{\ell=-m}^{m} a_\ell^{(2m)} g(\ell \boldsymbol{y})}|\boldsymbol{y}\rangle.$$

Using the relation between f and g, it is easy to see that the number of (fractional) phase queries to O_f we need in order to implement a modified oracle call $\mathrm{O}^S_{g_{(2m)}}$ is

$$\sum_{\ell=-m}^{m}\left\lceil\left|a_\ell^{(2m)}\right|S\right\rceil \le 2m + S\sum_{\ell=-m}^{m} a_\ell^{(2m)} \overset{(5.11)}{\le} 2m + S(2\log(m)+2). \tag{5.12}$$

Thus $\mathrm{O}^S_{g_{(2m)}}$ can be implemented using $\mathcal{O}\left(\frac{\log(m)}{\varepsilon r} + m\right)$ (fractional) queries to O_f. Picking $m = \log(c\sqrt{d}/\varepsilon)$ the query complexity becomes $\mathcal{O}\left(\frac{c\sqrt{d}}{\varepsilon} m \log(m)\right)$,[14] which is

$$\mathcal{O}\left(\frac{c\sqrt{d}}{\varepsilon}\log\left(\frac{c\sqrt{d}}{\varepsilon}\right)\log\log\left(\frac{c\sqrt{d}}{\varepsilon}\right)\right). \tag{5.13}$$ □

The above achieves, up to logarithmic factors, the desired $1/\varepsilon$ scaling in the precision parameter and also the $\sqrt{d}$ scaling with the dimension. This improves the results of [28] both quantitatively and qualitatively. We also show that the query complexity for this problem is almost optimal, by proving a lower bound in Section 6 which matches the above upper bound up to log factors.

5.5 Most quantum optimization problems are "smooth" We now show that the condition on the derivatives in Theorem 5.4 is fairly reasonable, i.e., a wide range of probability oracles that arise from quantum optimization problems satisfy this condition. In particular, consider the function $p : \mathbb{R}^d \to \mathbb{R}$ that we looked at (see Eq. (3.7)) during the discussion of a generic model of quantum optimization algorithms:

$$p(\boldsymbol{x}) = \langle \mathbf{0}|U(\boldsymbol{x})^\dagger(|1\rangle\langle 1|\otimes I)U(\boldsymbol{x})|\mathbf{0}\rangle.$$

We will now show that for every $k \in \mathbb{N}$ and index-sequence $\alpha \in [d]^k$, we have[15] $|\partial_\alpha p(\boldsymbol{x})| \le 2^k$ when $U(\boldsymbol{x})$ is a product of d (controlled) rotations given by

$$\mathrm{Rot}(x_j) = \exp\left[ix_j\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\right] = e^{ix_j\sigma_y}$$

and other fixed unitaries. In order to prove this, we first use Lemma 5.4 to show that $\|\partial_\alpha U(\boldsymbol{x})\| \le 1$, which by Lemma 5.5 implies that $\left\|\partial_\alpha\left(U(\boldsymbol{x})^\dagger(|1\rangle\langle 1|\otimes I)U(\boldsymbol{x})\right)\right\| \le 2^k$, hence proving the claim. In fact, we prove slightly stronger statements, so that these lemmas can be used later in greater generality.

LEMMA 5.4. *Suppose $\gamma \ge 0$ and*

$$U(\boldsymbol{x}) = U_0\prod_{j=1}^{d}\left(P_j\otimes e^{ix_jH_j} + (I-P_j)\otimes I\right)U_j,$$

where $\|U_0\| \le 1$ and for every $j \in [d]$ we a have that $\|U_j\| \le 1$, P_j is an orthogonal projection and H_j is Hermitian with $\|H_j\| \le \gamma$. Then for every $k \in \mathbb{N}$ and $\alpha \in [d]^k$, we have that $\|\partial_\alpha U(\boldsymbol{x})\| \le \gamma^k$.

Proof. We have that $\partial_\alpha U(\boldsymbol{x})$ equals

$$U_0\prod_{j=1}^{d}\Big(P_j\otimes(iH_j)^{|\{\ell\in[k]:\alpha_\ell=j\}|}e^{ix_jH_j} + 0^{|\{\ell\in[k]:\alpha_\ell=j\}|}(I-P_j)\otimes I\Big)U_j.$$

Therefore $\|\partial_\alpha U(\boldsymbol{x})\|$ can be upper bound by product of the norms of the individual terms in the product, hence

$$\|\partial_\alpha U(\boldsymbol{x})\| \le \prod_{j=1}^{d}\gamma^{|\{\ell\in[k]:\alpha_\ell=j\}|} = \gamma^k.$$ □

LEMMA 5.5. *Suppose that $A(\boldsymbol{x}), B(\boldsymbol{x})$ are linear operators parametrized by $\boldsymbol{x} \in \mathbb{R}^d$. If for all $k \in \mathbb{N}_0$ and $\alpha \in [d]^k$ we have that $\|\partial_\alpha A\| \le \gamma^k$ and $\|\partial_\alpha B\| \le \gamma^k$, then for all $k \in \mathbb{N}_0$ and $\alpha \in [d]^k$ we get that $\|\partial_\alpha(AB)\| \le (2\gamma)^k$.*

Proof. For an index-sequence $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k) \in [d]^k$ and a set $S = \{i_1 < i_2 < \ldots < i_\ell\} \subseteq [k]$ consisting of positions of the index-sequence, we define $\alpha_S := (\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_\ell}) \in [d]^{|S|}$ to be the index-sequence where we only keep indexes corresponding to positions in S; also let $\overline{S} := [k] \setminus S$. It can be seen that

$$\partial_\alpha(AB) = \sum_{S\subseteq[k]}\partial_{\alpha_S}A\partial_{\alpha_{\overline{S}}}B,$$

hence, $\|\partial_\alpha(AB)\|$ can be upper bounded by

$$\sum_{S\subseteq[k]}\left\|\partial_{\alpha_S}A\partial_{\alpha_{\overline{S}}}B\right\| \le \sum_{S\subseteq[k]}\gamma^{|S|}\gamma^{k-|S|} = (2\gamma)^k.$$ □

REMARK 5.1. *Finally note that the unitary in Lemma 5.4 is an analytic function of its parameters, therefore the probability that we get by taking products of such unitaries and some fixed matrices/vectors is also an analytic function.*

[14] If we strengthen the $c^k k^{\frac{k}{2}}$ upper bound assumption on the derivatives to c^k, then we could improve the bound of Theorem 5.3 by a factor of $k^{-k/2}$. Therefore in the definition of R^{-1} we could replace m by $\sqrt{m}$ which would quadratically improve the log factor in (5.13).

[15] This essentially means that the function $p(\boldsymbol{x})$ in the probability oracle is in the Gevrey class G^0.

6 Lower bounds on gradient computation

In this section we prove that the number of phase oracle queries required to compute the gradient for some of the smooth functions satisfying the requirement of Theorem 5.4 is $\Omega\left(\sqrt{d}/\varepsilon\right)$, showing that Theorem 5.4 is optimal up to log factors. As we show in the full version [22], a probability oracle can be simulated using a logarithmic number of phase oracle queries, therefore this lower bound also translates to probability oracles.[16]

In the full version of this paper we prove Theorem 1.2 providing a lower bound on the number of queries needed in order to distinguish an unknown phase oracle from a family of phase oracles. This result is of independent interest, and has found an application for proving lower bounds on quantum SDP-solving [2].

Using Theorem 1.2 we prove our lower on gradient computation by constructing a family of functions which can be distinguished from the trivial phase oracle I by ε-precise gradient computation, but for which our hybrid-method based lower bound shows that distinguishing the phase oracles from I requires $\Omega\left(\sqrt{d}/\varepsilon\right)$ queries.

6.1 A family of functions for proving the gradient computation lower bound

Now we prove our lower on gradient computation by constructing a family of functions $\mathcal{F}$ for which the corresponding phase oracles $\{\mathrm{O}_f : f \in \mathcal{F}\}$ require $\Omega(\sqrt{d}/\varepsilon)$ queries to distinguish them from the constant 0 function (as shown by Theorem 1.2), but the functions in $\mathcal{F}$ can be uniquely identified by calculating their gradient at $\mathbf{0}$ with accuracy ε. In particular, this implies that calculating an approximation of the gradient vector for these functions must be at least as hard as distinguishing the phase oracles corresponding to functions in $\mathcal{F}$.

LEMMA 6.1. *Let $d \in \mathbb{N}$, $\varepsilon, c \in \mathbb{R}_+$ and let us define the following $\mathbb{R}^d \to \mathbb{R}$ functions: $f_*(\boldsymbol{x}) := 0$ and $f_j(\boldsymbol{x}) := 2\varepsilon x_j e^{-c^2\|\boldsymbol{x}\|^2/2}$ for all $j \in [d]$. Consider the family of functions $\mathcal{F} := \bigcup_{j\in[d]}\{f_j(\boldsymbol{x})\}$, then for all $\boldsymbol{x} \in \mathbb{R}^d$ we have that*

$$\sum_{j\in[d]} |f_j(\boldsymbol{x}) - f_*(\boldsymbol{x})|^2 \le \frac{4\varepsilon^2}{ec^2}.$$

Proof.

$$\sum_{j\in[d]} |f_j(\boldsymbol{x}) - f_*(\boldsymbol{x})|^2 = \sum_{j\in[d]} \left|2\varepsilon x_j e^{-c^2\|\boldsymbol{x}\|^2/2}\right|^2 = 4\varepsilon^2\|\boldsymbol{x}\|^2 e^{-c^2\|\boldsymbol{x}\|^2} \le \frac{4\varepsilon^2}{ec^2},$$

where we used $ze^{-z} \le 1/e$ with $z := c^2\|\boldsymbol{x}\|^2$. □

In the full version of this paper [22] we prove bounds on the partial derivatives of the above functions to determine their smoothness, resulting in the following lemma.

LEMMA 6.2. *Let d, k be positive integers, $c \in \mathbb{R}_+$ and $\boldsymbol{x} \in \mathbb{R}^d$. Then, the function $f_j(\boldsymbol{x}) := cx_j e^{-\frac{c^2\|\boldsymbol{x}\|^2}{2}}$ satisfies the following: for every index-sequence $\alpha \in [d]^k$, the derivative of f is bounded by $|\partial_\alpha f(\mathbf{0})| \le c^k k^{\frac{k}{2}}$. Moreover $\nabla f_j(\mathbf{0}) = c\boldsymbol{e}_j$.*

Now use the above lemmas combined with the hybrid method Theorem 1.2 to prove our general lower bound result which implies the earlier stated informal lower bound of Theorem 1.1.

THEOREM 6.1. *Let $\varepsilon, c, d > 0$ such that $2\varepsilon \le c$ and for an arbitrary finite set $G \subseteq \mathbb{R}^d$ let*

$$\mathcal{H} = \operatorname*{Span}_{\boldsymbol{x}\in G}\{|\boldsymbol{x}\rangle : \boldsymbol{x} \in G\}.$$

Suppose $\mathcal{A}$ is a T-query quantum algorithm (assuming query access to phase oracle $\mathrm{O}_f : |\boldsymbol{x}\rangle \mapsto e^{if(\boldsymbol{x})}$, acting on $\mathcal{H}$) for analytic functions $f : \mathbb{R}^d \to \mathbb{R}$ satisfying

$$|\partial_\alpha f(\mathbf{0})| \le c^k k^{\frac{k}{2}} \quad \text{for all } k \in \mathbb{N}, \alpha \in [d]^k,$$

such that $\mathcal{A}$ computes an ε-approximation $\tilde{\nabla} f(\mathbf{0})$ of the gradient at $\mathbf{0}$ (i.e., $\left\|\tilde{\nabla} f(\mathbf{0}) - \nabla f(\mathbf{0})\right\|_\infty < \varepsilon$), succeeding with probability at least $2/3$. Then $T > \frac{c\sqrt{d}}{4\varepsilon}$.

Proof. Inspired by Lemma 6.1, we first define a set of "hard" functions, which we will use to prove our lower bound Let $f_* := f_0 := 0$ and $f_j(\boldsymbol{x}) := 2\varepsilon x_j e^{-c^2\|\boldsymbol{x}\|^2/2}$ for all $j \in [d]$. Consider the family of functions $\mathcal{F} := \bigcup_{j\in[d]}\{f_j(\boldsymbol{x})\}$. By Lemma 6.2, every $f \in \mathcal{F}$ satisfies $|\partial_\alpha f(\mathbf{0})| \le c^k k^{\frac{k}{2}}$ for all $k \in \mathbb{N}$ and $\alpha \in [d]^k$.

Suppose we are given a phase oracle O_f acting on $\mathcal{H}$, such that $\mathrm{O}_f = \mathrm{O}_{f_j} : |\boldsymbol{x}\rangle \mapsto e^{if_j(\boldsymbol{x})}|\boldsymbol{x}\rangle$ for an $j \in \{0, \dots, d\}$. Since $\nabla f_0(\mathbf{0}) = \mathbf{0}$ and $\nabla f_j(\mathbf{0}) = 2\varepsilon\boldsymbol{e}_j$, using the T-query algorithm $\mathcal{A}$ in the theorem statement, one can determine the $j \in \{0, \dots, d\}$ for which $f_j = f$ with success probability at least $2/3$. In particular we can distinguish the case $f = f_*$ from $f \in \mathcal{F}$, and thus by Theorem 1.2 and Lemma 6.1 we get that

$$T \ge \sqrt{d}\frac{c}{\varepsilon}\sqrt{\frac{e}{36}} > \frac{c\sqrt{d}}{4\varepsilon}.$$

□

[16] A probability oracle can only represent functions which map to $[0, 1]$, whereas the range of the function f we use in the lower bound proof is an subinterval of $[-1, 1]$. However, by using the transformed function $g := (2+f)/4$ we get a function which has a range contained in $[1/2, 3/4]$ so it can in principle be represented by a probability oracle. Moreover for a function with a range contained in $[1/2, 3/4]$ we can efficiently convert between phase and probability oracles as shown in the full version [22].

6.2 Lower bound for more regular functions

Note that the functions for which we apply our results in this paper tend to satisfy a stronger condition than our lower bound example in Theorem 6.1. They usually satisfy[17] $|\partial_\alpha f(\boldsymbol{x}_0)| \leq c^k$. We conjectured that the same lower bound holds for this subclass of functions as well.

Very recently, Cornelissen [16] managed to prove this conjecture (also building on our version of the hybrid-method Theorem 1.2). Moreover, he showed an $\Omega\left(d^{\frac{1}{2}+\frac{1}{p}}/\varepsilon\right)$ lower bound for ε-precise gradient computation in p-norm *for every* $p \in [1, \infty]$. This lower bound matches our upper bound[18] up to logarithmic factors, showing that our algorithm is essentially optimal for a large class of gradient computation problems.

7 Applications

In this section, we first consider variational quantum eigensolvers and QAOA algorithms, which can be treated essentially identically using our techniques. Then we consider the training of quantum autoencoders, which requires a slightly different formalism. We show that our gradient descent algorithms can be applied to these problems by reducing such problems to a probability maximization problem. For each application our quantum gradient computation algorithm yields a quadratic speedup in terms of the dimension.

7.1 Variational quantum eigensolvers

In recent years, variational quantum eigensolvers and QAOA [39, 47, 19] are favored methods for providing low-depth quantum algorithms for solving important problems in quantum simulation and optimization. Current quantum computers are limited by decoherence, hence the option to solve optimization problems using very short circuits can be enticing even if such algorithms are polynomially more expensive than alternative strategies that could possibly require long gate sequences. Since these methods are typically envisioned as being appropriate only for low-depth applications, comparably less attention is paid to the question of what their complexity would be, if they were executed on a fault-tolerant quantum computer. In this paper, we consider the case that these algorithms are in fact implemented on a fault-tolerant quantum computer and show that the gradient computation step in these algorithms can be performed quadratically faster compared to the earlier approaches that were tailored for pre-fault-tolerant applications.

[17] Without the $k^{\frac{k}{2}}$ factor – i.e., they are of Gevrey class G^0 instead of $G^{\frac{1}{2}}$.

[18] If we want an ε-approximate gradient in p-norm, it suffices to compute an $(\varepsilon d^{-\frac{1}{p}})$-approximate gradient in the ∞-norm due to Hölder' inequality.

Variational quantum eigensolvers (VQEs) are widely used to estimate the eigenvalue corresponding to some eigenstate of a Hamiltonian. The idea behind these approaches is to begin with an efficiently parameterizable ansatz to the eigenstate. For the example of ground state energy estimation, the ansatz state is often taken to be a unitary coupled cluster expansion. The terms in that unitary coupled cluster expansion are then varied to provide the lowest energy for the groundstate. For excited states a similar argument can be applied, but minimizing a free-energy rather than ground state energy is the most natural approach.

For simplicity, let us focus on the simplest (and most common) example of groundstate estimation. Consider a Hamiltonian of the form $H = \sum_j a_j U_j$ where U_j is a unitary matrix, $a_j > 0$ and $\sum_j a_j = 1$. This assumption can be made w.l.o.g by renormalizing the Hamiltonian and absorbing signs into the unitary matrix. Let the state $|\psi(\boldsymbol{x})\rangle$ for $\boldsymbol{x} \in \mathbb{R}^d$ be the variational state prepared by the Prep. and Tuned circuits in Fig. 1b. Our objective function is then to estimate

$$\boldsymbol{x}_{\text{opt}} = \operatorname*{argmin}_{\boldsymbol{x}} \left(\langle \psi(\boldsymbol{x})| \sum_j a_j U_j |\psi(\boldsymbol{x})\rangle \right), \tag{7.14}$$

which is real valued because H is Hermitian.

In order to translate this problem to one that we can handle using our gradient descent algorithm, we construct a verifier circuit that given $|\psi(\boldsymbol{x})\rangle$ sets an ancilla qubit to 1 with probability $p = (1 + \langle\psi(\boldsymbol{x})|H|\psi(\boldsymbol{x})\rangle)/2$. This is possible since $\|H\| \leq 1$ due to the assumption that $\sum_j a_j = 1$. This motivates the definition of new input oracles used for implementing the Hamiltonian.

$$\text{prepareW}\colon |0\rangle \mapsto \sum_j \sqrt{a_j}|j\rangle, \tag{7.15}$$

$$\text{selectH} := \sum_j |j\rangle\langle j| \otimes U_j. \tag{7.16}$$

We can then use the unitaries (7.15)-(7.16) to define and compute the query complexity of performing a single variational step in a VQE algorithm.

COROLLARY 7.1. *Let* $\text{Tuned}(\boldsymbol{x}) = \prod_{j=1}^d e^{-iH_j x_j}$ *for* $\|H_j\| = 1$ *and* $\boldsymbol{x} \in \mathbb{R}^d$ *and let* $\text{Prep} = I$. *If* $H = \sum_{j=1}^M a_j H_j$ *for unitary* H_j *with* $a_j \geq 0$ *and* $\sum_j a_j = 1$ *then the number of queries to prepareW, selectH and Tuned needed to output a qubit string* $|\boldsymbol{y}\rangle$ *such that* $|\nabla\langle\psi(\boldsymbol{x})|H|\psi(\boldsymbol{x})\rangle - \boldsymbol{y}| \leq \varepsilon$ *with probability at least* $2/3$ *is* $\widetilde{\mathcal{O}}\left(\sqrt{d}/\varepsilon\right)$.

Proof. First we argue that the circuit in Fig. 2 outputs the claimed probability. We then convert this into

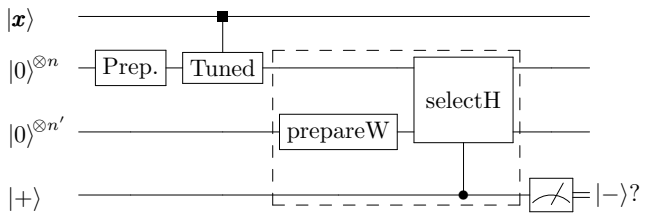


Figure 2: Circuit for converting groundstate energy to a probability for VQE. The dashed box denotes the verifier circuit, V, in Fig. 1b which is implemented here using the Hadamard test. (To get the actual Hadamard test circuit, one should add a prepareW† gate. This gate is in fact unnecessary therefore we omitted it.) Probability of measuring 1 is $1/2 - \langle\psi(\boldsymbol{x})|H|\psi(\boldsymbol{x})\rangle/2$. In our description the Prep. circuit is just the identity gate, but in general it might be useful to apply some preprocessing.

a phase oracle, use our improvement over Jordan's algorithm (Theorem 5.4) and prove that $c \in \mathcal{O}(1)$ for this problem to show the claimed complexity.

First, if we examine the gates in Fig. 2 we note that the prep and Tuned oracles by definition prepare the state $|\psi(\boldsymbol{x})\rangle$. In this context the prep circuit is the identity. While this could be trivially removed from the circuit, we retain it to match the formal description of the model that we give earlier. Under the assumption that $\sum_j a_j = 1$ note that

$$\langle 0|\langle\psi(\boldsymbol{x})|\text{prepareW}^\dagger(\text{selectH})\text{prepareW}|0\rangle|\psi(\boldsymbol{x})\rangle \tag{7.17}$$
$$= \sum_j \sum_k \sqrt{a_j a_k}\langle k|j\rangle \otimes \langle\psi(\boldsymbol{x})|U_j|\psi(\boldsymbol{x})\rangle.$$

$$= \langle\psi(\boldsymbol{x})| \sum_j a_j U_j|\psi(\boldsymbol{x})\rangle = \langle\psi(\boldsymbol{x})|H|\psi(\boldsymbol{x})\rangle. \tag{7.18}$$

Then because prepareW is unitary it follows that controlling the selectH operation enacts the controlled prepareW†(selectH)prepareW operation.

The claim regarding the probability then follows directly from the Hadamard test, which we prove below for completeness. Let $\Lambda(U)$ be a controlled unitary operation. Then $H\Lambda(U)H|0\rangle|\psi(\boldsymbol{x})\rangle$ equals

$$H(|0\rangle|\psi(\boldsymbol{x})\rangle + |1\rangle U|\psi(\boldsymbol{x})\rangle)/\sqrt{2}.$$
$$= |0\rangle\left(\frac{(1+U)|\psi(\boldsymbol{x})\rangle}{2}\right) + |1\rangle\left(\frac{(1-U)|\psi(\boldsymbol{x})\rangle}{2}\right) \tag{7.19}$$

Thus it follows from Born's rule that the probability of measuring the first register to be 1 is $(1 - \text{Re}(\langle\psi|U|\psi\rangle))/2$. We then have from combining this result with (7.18) and recalling that H is Hermitian gives us that the probability of measuring 1 in the output of the circuit in Fig. 2 is $1/2 - \langle\psi|H|\psi\rangle/2$ as claimed. Thus we have an appropriate probability oracle for the approximate groundstate energy expectation.

Each query to the circuit of Fig. 2 requires $\mathcal{O}(1)$ queries to prepareW and selectH. Thus the probability oracle can be simulated at cost $\mathcal{O}(1)$ fundamental queries. Now if we remove the measurement from the circuit we see that we can view the transformation as a circuit of the form

$$\begin{aligned} U|0\rangle =& \sqrt{1/2 - \langle\psi(\boldsymbol{x})|H|\psi(\boldsymbol{x})\rangle/2}|\psi_{\text{good}}\rangle|1\rangle \\ &+ \sqrt{1/2 + \langle\psi(\boldsymbol{x})|H|\psi(\boldsymbol{x})\rangle/2}|\psi_{\text{bad}}\rangle|0\rangle. \end{aligned} \tag{7.20}$$

that for any $\delta \in (0, 1/3)$ we can simulate a δ–approximate query to the phase oracle analogue of U using $\mathcal{O}(\log(1/\delta))$ applications of U. Since U requires $\mathcal{O}(1)$ fundamental queries, the phase oracle can be implemented using $\mathcal{O}(\log(1/\delta))$ fundamental queries to selectH, prepareW, Tuned and Prep. From Theorem 5.4 it then follows that we can compute

$$\nabla(1/2 - \langle\psi(\boldsymbol{x})|H|\psi(\boldsymbol{x})\rangle/2) = -\nabla\langle\psi(\boldsymbol{x})|H|\psi(\boldsymbol{x})\rangle/2,$$

within error $\varepsilon/2$ and error probability bounded above by $1/3$ using $\widetilde{\mathcal{O}}\left(c\sqrt{d}/\varepsilon\right)$ applications of selectH and prepareW. Our result then immediately follows if $c \in \widetilde{\mathcal{O}}(1)$. This is equivalent to proving that for some $c \in \widetilde{\mathcal{O}}(1)$ we have that $|\partial_{\alpha_1}\cdots\partial_{\alpha_k}\langle\psi|H|\psi\rangle| \le c^k$ holds for all $k \in \mathbb{N}$ and $\alpha \in [d]^k$.

We prove that for this application $c \le 2$. To see this note that for any index sequence $\alpha \in [d]^k$, we can upper bound $|\partial_\alpha\langle\psi(\boldsymbol{x})|H|\psi(\boldsymbol{x})\rangle|$ by

$$\begin{aligned} &\left\|\partial_\alpha\left(\prod_{j=d}^{1} e^{iH_j x_j} H \prod_{j=1}^{d} e^{-iH_j x_j}\right)\right\| \\ &\le \sum_{\ell=1}^{M} |a_\ell| \left\|\partial_\alpha\left(\prod_{j=d}^{1} e^{iH_j x_j} H_\ell \prod_{j=1}^{d} e^{-iH_j x_j}\right)\right\|. \end{aligned} \tag{7.21}$$

Lemma 5.4 directly implies that

$$\left\|\partial_\alpha \prod_{j=d}^{1} e^{iH_j x_j}\right\| = 1, \tag{7.22}$$

and since H_ℓ is unitary and Hermitian for all ℓ we have

$$\left\|H_\ell \partial_\alpha \prod_{j=1}^{d} e^{-iH_j x_j}\right\| = 1. \tag{7.23}$$

Finally, Lemma 5.5 and Eq. (7.22),(7.23) implies

$$\sum_{\ell=1}^{M} |a_\ell| \left\| \partial_\alpha \left(\prod_{j=d}^{1} e^{iH_j x_j} H_\ell \prod_{j=1}^{d} e^{-iH_j x_j} \right) \right\| \leq \sum_{\ell=1}^{M} |a_\ell| 2^k = 2^k. \qquad \square$$

To compare the complexity for this procedure in an unambiguous way to that of existing methods, we need to consider a concrete alternative model for the cost. For classical methods, we typically assume that the a_j are known classically as are the U_j that appear from queries to selectH. For this reason, the most relevant aspect to compare is the number of queries needed to the Tuned oracle. The number of state preparations needed to estimate the gradient is clearly $\widetilde{\mathcal{O}}(d/\varepsilon^2)$ using high-degree gradient methods on the empirically estimated gradients [47] if we assume that the gradient needs to be computed with constant error in $\|\cdot\|_\infty$. In this sense, we provide a quadratic improvement over such methods when the selectH and prepareW oracles are sub-dominant to the cost of the state preparation algorithm.

The application of this method to QAOA directly follows from the analysis given above. There are many flavors of the quantum approximate optimization algorithm (QAOA) [19]. The core idea of the algorithm is to consider a parametrized family of states such as $|\psi(\boldsymbol{x})\rangle = \prod_{j=1}^{d} e^{-ix_j H_j}|0\rangle$. The aim is to modify the state in such a way as to maximize an objective function. In particular, if we let O be a Hermitian operator corresponding to the objective function then we wish to find $\boldsymbol{x}$ such that $\langle\psi(\boldsymbol{x})|H|\psi(\boldsymbol{x})\rangle$ is maximized. For example, in the case of combinatorial optimization problems the objective function is usually expressed as the number of satisfied clauses: $O = \sum_{\alpha=1}^{m} C_\alpha$ where C_α is 1 if and only if the α^{th} clause is satisfied and 0 otherwise [19]. Such clauses can be expressed as sums of tensor products of Pauli operators, which allows us to express them as Hermitian operators. Thus, from the perspective of our algorithm, QAOA looks exactly like variational quantum eigensolvers except in that the parameterization chosen for the state may be significantly different from that chosen for variational quantum eigensolvers.

7.2 Quantum auto-encoders Classically, one application of neural networks is *auto-encoders*, which are networks that encode information about a data set into a low-dimensional representation. Auto-encoding was first introduced by Rumelhart et al. [42]. Informally, the goal of an auto-encoding circuit is the following: suppose we are given a set of high-dimensional vectors, we would like to learn a representation of the vectors hopefully of low dimension, so that computations on the original data set can be "approximately" carried out by working only with the low-dimensional vectors. More precisely the problem in auto-encoding is: Given $K < N$ and m data vectors $\{v_1, \ldots, v_m\} \subseteq \mathbb{R}^N$, find an encoding map $\mathcal{E} : \mathbb{R}^N \to \mathbb{R}^K$ and decoding map $\mathcal{D} : \mathbb{R}^K \to \mathbb{R}^N$ such that the average squared distortion $\|v_i - (\mathcal{D} \circ \mathcal{E})(v_i)\|^2$ is minimized:[19]

$$\min_{\mathcal{E},\mathcal{D}} \sum_{i\in[m]} \frac{\|v_i - (\mathcal{D} \circ \mathcal{E})(v_i)\|^2}{m}. \tag{7.24}$$

What makes auto-encoding interesting is that it does not assume any prior knowledge about the data set. This makes it a viable technique in machine learning, with various applications in natural language processing, training neural networks, object classification, prediction or extrapolation of information, etc.

Given that classical auto-encoders are 'work-horses' of classical machine learning [5], it is also natural to consider a quantum variant of this paradigm. Very recently such quantum auto-encoding schemes have been proposed by Wan Kwak et al. [45] and independently by Romero et al. [41]. Inspired by their work we provide a slightly generalized description of quantum auto-encoders by 'quantizing' auto-encoders the following way: we replace the data vectors v_i by quantum states ρ_i and define the maps $\mathcal{E}, \mathcal{D}$ as quantum channels transforming states back and forth between the Hilbert spaces $\mathcal{H}$ and $\mathcal{H}'$. A natural generalization of squared distortion for quantum states ρ, σ that we consider is $1 - F^2(\rho, \sigma)$,[20] giving us the following minimization problem

$$\min_{\mathcal{E},\mathcal{D}} \sum_{i\in[m]} \frac{1 - F^2(\rho_i, (\mathcal{D} \circ \mathcal{E})(\rho_i))}{m}. \tag{7.25}$$

Since $F^2(|\psi\rangle\langle\psi|, \sigma) = \langle\psi|\sigma|\psi\rangle$ in the special case when the input states are pure states $\rho_i = |\psi_i\rangle\langle\psi_i|$, the above minimization problem is equivalent to the maximization problem

$$\max_{\mathcal{E},\mathcal{D}} \sum_{i\in[N]} \frac{\langle\psi_i|[(\mathcal{D} \circ \mathcal{E})(|\psi_i\rangle\langle\psi_i|)]|\psi_i\rangle}{m}. \tag{7.26}$$

[19] There are other natural choices of dissimilarity functions that one might want to minimize, for a comprehensive overview of the classical literature see [6].

[20] Note that some authors (including [41]) call $F' = F^2$ the fidelity. The distortion measure we use here is $P(\rho, \sigma) = \sqrt{1 - F^2(\rho, \sigma)}$, which is called the purified (trace) distance [44].

Observe that $\langle\psi|[(\mathcal{D}\circ\mathcal{E})(|\psi\rangle\langle\psi|)]|\psi\rangle$ is the probability of finding the output state $(\mathcal{D}\circ\mathcal{E})(|\psi\rangle\langle\psi|)$ in state $|\psi\rangle$ after performing the projective measurement $\{|\psi\rangle\langle\psi|, I - |\psi\rangle\langle\psi|\}$. Thus we can think about this as maximizing the probability of recovering the initial pure state after encoding and decoding, which is a natural measure of the quality of the quantum auto-encoding procedure.

7.2.1 Training quantum auto-encoders Similarly to [45, 41] we describe a way to perform this optimization problem in the special case when the input states are n-qubit pure states and they are mapped to k-qubit states, i.e., $\mathcal{H}$ is the Hilbert space of n qubits and $\mathcal{H}'$ is the Hilbert space of $k < n$ qubits. We also show how our gradient computation algorithm can speedup solving the described optimization problem.

We observe that by adding a linear amount of ancilla qubits we can represent the encoding and decoding channels by unitaries, which makes the minimization conceptually simpler. Indeed by Stinespring's dilation theorem [46, Corollary 2.27], [32] we know that any quantum channel $\mathcal{E}$ that maps n qubit states to k qubit states can be constructed by adding $2k$ qubits initialized in $|\vec{0}\rangle$ state, then acting with a unitary $U_{\mathcal{E}}$ on the extended space and then tracing out $k+n$ qubits. Applying this result to both $\mathcal{E}$ and $\mathcal{D}$ results in a unitary circuit representing the generic encoding/decoding procedure, see Figure 3. (This upper bound on the required number of ancilla qubits for $\mathcal{D}$ becomes $2n$.)

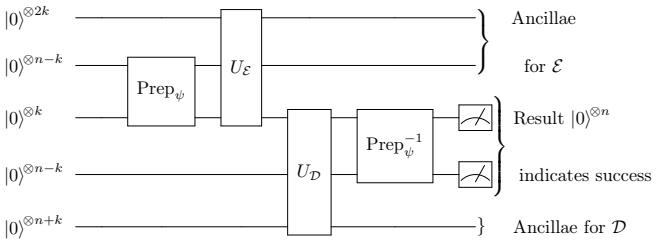


Figure 3: A unitary quantum auto-encoding circuit: For the input $|\psi\rangle$, the circuit prepares $|\psi\rangle$, applies a purified version of the channels $\mathcal{E}, \mathcal{D}$ and finally checks by a measurement whether the decoded state is $|\psi\rangle$.

In order to solve the maximization problem (7.26) we could just introduce a parametrization of the unitaries $U_{\mathcal{E}}, U_{\mathcal{D}}$ and search for the optimal parameters using gradient descent. Unfortunately a complete parametrization of the unitaries requires exponentially many parameters, which is prohibitive. However, analogously to, e.g., classical machine learning practices, one could hope that a well-structured circuit can achieve close to optimal performance using only a polynomial number of parameters. If the circuits $U_{\mathcal{E}}, U_{\mathcal{D}}$ are parametrized nicely, so that Lemma 5.4 can be applied, then we can use our gradient computation algorithm to speedup optimization.

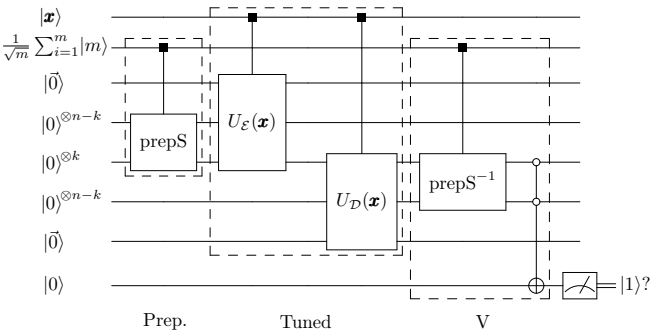


Figure 4: Quantum circuit which outputs 1 with probability equal to the objective function (7.26). The structure of the circuit fits the generic model of quantum optimization circuits (Figure 1), therefore we can use our gradient computation methods to speedup its optimization.

We can do the whole optimization using stochastic gradient descent [27], so that in each step we only need to consider the effect of the circuit on a single pure state. Or if we have more quantum resources available we can directly evaluate the full gradient by preparing a uniform superposition over all input vectors. In this case the state preparation unitary $\mathrm{Prep} = \sum_{i=1}^{m} |i\rangle\langle i| \otimes \mathrm{Prep}_{\psi_i}$ is a controlled unitary, which controlled on index i would prepare $|\psi_i\rangle$. Graphically we represent this type of control by a small black square in contrast to the small black circle used for denoting simple controlled unitaries. See the full quantum circuit in Figure 4.

Finally, note that in some application it might be desirable to ask for a coherent encoding/decoding procedure, where all the ancilla qubits are returned to the $|\vec{0}\rangle$ state. In this case similarly to [45, 41] one could define $U_{\mathcal{D}} = U_{\mathcal{E}}^{-1}$ and optimize the probability of measuring $|\vec{0}\rangle$ on the ancilla qubits after applying $U_{\mathcal{E}}$.

8 Conclusion and future research

We gave a new approach to quantum gradient computation that is asymptotically optimal (up to logarithmic factors) for a class of smooth functions, in terms of the number of queries needed to estimate the gradient within fixed error with respect to the max-norm. This is based on several new ideas including the use of differentiation formulæ originating from high-degree interpolatory polynomials. These high-degree methods quadratically improve the scaling of the query complexity with respect to the approximation quality compared to what one would see if the results from Jordan's work were used. In the case of low-degree multivariate polynomials we showed that our algorithm can yield an ex-

ponential speedup compared to Jordan's algorithm or classical algorithms. We also provided lower bounds on the query complexity of the problem for certain smooth functions revealing that our algorithm is essentially optimal for a class of functions.

While it has proven difficult to find natural applications for Jordan's original algorithm, we provide in this paper several applications of our gradient descent algorithm to areas ranging from machine learning to quantum chemistry simulation. These applications are built upon a method we provide for interconverting between phase and probability oracles. The polynomial speedups that we see for these applications is made possible by our improved quantum gradient algorithm via the use of this interconversion process. It would be interesting to find applications where we can apply the results for low-degree multivariate polynomials providing an exponential speedup.

More work remains to be done in developing quantum techniques that speed up more sophisticated higher-level, e.g., stochastic gradient descent methods. Another interesting question is whether quantum techniques can provide further speedups for calculating higher-order derivatives, such as the Hessian, using ideas related to Jordan's algorithm, see e.g. [29, Appendix D]. Such improvements might open the door for improved quantum analogues of Newton's method and in turn substantially improve the scaling of the number of epochs needed to converge to a local optima.

Acknowledgements

The authors thank Ronald de Wolf, Arjan Cornelissen, Yimin Ge and Robin Kothari for helpful suggestions and discussions. A.G. and N.W. thank Sebastien Bubeck, Michael Cohen and Yuanzhi Li for insightful discussions about accelerated gradient methods. A.G. thanks Joran van Apeldoorn for drawing his attention to the "cheap gradient principle".

References

[1] A. Ambainis, Variable time amplitude amplification and quantum algorithms for linear algebra problems, in STACS, 2012, pp. 636–647. arXiv: 1010.4458

[2] J. v. Apeldoorn and A. Gilyén, Improvements in quantum SDP-solving with applications. arXiv: 1804.05058, 2018.

[3] J. v. Apeldoorn, A. Gilyén, S. Gribling, and R. d. Wolf, Quantum SDP-solvers: Better upper and lower bounds, in FOCS, 2017, pp. 403–414. arXiv: 1705.01843

[4] ———, Convex optimization using quantum oracles. arXiv: 1809.00643, 2018.

[5] E. M. Azoff, *Neural Network Time Series Forecasting of Financial Markets*, John Wiley & Sons, New York, NY, USA, 1st ed., 1994.

[6] P. Baldi, Autoencoders, unsupervised learning, and deep architectures, in Unsupervised and Transfer Learning - Workshop held at ICML 2011, 2012, pp. 37–50.

[7] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, Strengths and weaknesses of quantum computing, SIAM J. Comp., 26 (1997), pp. 1510–1523. arXiv: quant-ph/9701001

[8] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan, Time bounds for selection, J. Comp. Sys. Sci., 7 (1973), pp. 448–461.

[9] F. G. S. L. Brandão, A. Kalev, T. Li, C. Y.-Y. Lin, K. M. Svore, and X. Wu, Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning. arXiv: 1710.02581, 2017.

[10] F. G. S. L. Brandão and K. M. Svore, Quantum speed-ups for solving semidefinite programs, in FOCS, 2017, pp. 415–426. arXiv: 1609.05537

[11] D. Bulger, Quantum basin hopping with gradient-based local optimisation. Unpublished. arXiv: quant-ph/0507193, 2005.

[12] ———, Quantum computational gradient estimation. Unpublished. arXiv: quant-ph/0507109, 2005.

[13] S. Chakrabarti, A. M. Childs, T. Li, and X. Wu, Quantum algorithms and lower bounds for convex optimization. arXiv: 1809.01731, 2018.

[14] A. M. Childs, R. Kothari, and R. D. Somma, Quantum algorithm for systems of linear equations with exponentially improved dependence on precision, SIAM J. Comp., 46 (2017), pp. 1920–1950. arXiv: 1511.02306

[15] R. Cleve, D. Gottesman, M. Mosca, R. D. Somma, and D. L. Yonge-Mallo, Efficient discrete-time simulations of continuous-time quantum query algorithms, in STOC, 2009, pp. 409–416. arXiv: 0811.4428

[16] A. Cornelissen, Quantum gradient estimation and its application to quantum reinforcement learning, Master's thesis, Technische Universiteit Delft, 2018.

[17] C. Dürr and P. Høyer, A quantum algorithm for finding the minimum. arXiv: quant-ph/9607014, 1996.

[18] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla, Quantum query complexity of some graph problems, SIAM J. Comp., 35 (2006), pp. 1310–1328. arXiv: quant-ph/0401091

[19] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm. arXiv: 1411.4028, 2014.

[20] E. Farhi and H. Neven, Classification with quantum neural networks on near term processors. arXiv: 1802.06002, 2018.

[21] M. Gevrey, Sur la nature analytique des solutions des équations aux dérivées partielles. Premier mémoire., Annales Scientifiques de l'École Normale Supérieure,

3 (1918), pp. 129–190.
[22] A. Gilyén, S. Arunachalam, and N. Wiebe, Optimizing quantum optimization algorithms via faster quantum gradient computation. arXiv: 1711.00465v3. Full version of this paper, 2017.
[23] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. arXiv: 1806.01838, 2018.
[24] L. K. Grover, A fast quantum mechanical algorithm for database search, in STOC, 1996, pp. 212–219. arXiv: quant-ph/9605043
[25] Y. Hamoudi and F. Magniez, Quantum Chebyshev's inequality and applications. arXiv: 1807.06456, 2018.
[26] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett., 103 (2009), p. 150502. arXiv: 0811.3171
[27] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford, Accelerating stochastic gradient descent. arXiv: 1704.08227, 2017.
[28] S. P. Jordan, Fast quantum algorithm for numerical gradient estimation, Phys. Rev. Lett., 95 (2005), p. 050501. arXiv: quant-ph/0405146
[29] ———, *Quantum Computation Beyond the Circuit Model*, PhD thesis, Massachusetts Institute of Technology, 2008. arXiv: 0809.2307
[30] I. Kerenidis and A. Prakash, Quantum gradient descent for linear systems and least squares. arXiv: 1704.04992, 2017.
[31] ———, A quantum interior point method for LPs and SDPs. arXiv: 1808.09266, 2018.
[32] M. Keyl, Fundamentals of quantum information theory, Phys. Rep., 369 (2002), pp. 431 – 548. arXiv: quant-ph/0202122
[33] P. C. S. Lara, R. Portugal, and C. Lavor, A new hybrid classical-quantum algorithm for continuous global optimization problems, J. Global Optimization, 60 (2014), pp. 317–331. arXiv: 1301.4667
[34] J. Li, General explicit difference formulas for numerical differentiation, Journal of Computational and Applied Mathematics, 183 (2005), pp. 29 – 52.
[35] G. H. Low and I. L. Chuang, Hamiltonian simulation by qubitization. arXiv: 1610.06546, 2016.
[36] A. Montanaro, Quantum speedup of Monte Carlo methods, Proc. Roy. Soc. A, 471 (2015). arXiv: 1504.06987
[37] A. Nayak and F. Wu, The quantum query complexity of approximating the median and related statistics, in STOC, 1999, pp. 384–393. arXiv: quant-ph/9804066
[38] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*, Cambridge University Press, 2000.
[39] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Comm., 5 (2014). arXiv: 1304.3061
[40] P. Rebentrost, M. Schuld, F. Petruccione, and S. Lloyd, Quantum gradient descent and Newton's method for constrained polynomial optimization. arXiv: 1612.01789, 2016.
[41] J. Romero, J. P. Olson, and A. Aspuru-Guzik, Quantum autoencoders for efficient compression of quantum data, Quantum Science and Technology, 2 (2017), p. 045001. arXiv: 1612.02806
[42] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning internal representations by error propagation, in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, D. E. Rumelhart, J. L. McClelland, and P. R. Group, eds., MIT Press, Cambridge, MA, USA, 1986, pp. 318–362.
[43] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, Circuit-centric quantum classifiers. arXiv: 1804.00633, 2018.
[44] M. Tomamichel, R. Colbeck, and R. Renner, Duality between smooth min- and max-entropies, IEEE Transactions on Information Theory, 56 (2010), pp. 4674–4681. arXiv: 0907.5238
[45] O. D. Wan, Kwok Ho, H. Kristjánsson, R. Gardner, and M. S. Kim, Quantum generalisation of feedforward neural networks. arXiv: 1612.01045, 2016.
[46] J. Watrous, *The Theory of Quantum Information*, Cambridge University Press, 2018.
[47] D. Wecker, M. B. Hastings, and M. Troyer, Progress towards practical quantum variational algorithms, Phys. Rev. A, 92 (2015), p. 042303. arXiv: 1507.08969
[48] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum nearest-neighbor algorithms for machine learning, Quant. Inf. & Comp., 15 (2015), pp. 318–358. arXiv: 1401.2142