Quantum algorithms and lower bounds for convex optimization

Shouvanik Chakrabarti, Andrew M. Childs, Tongyang Li, and Xiaodi Wu

Department of Computer Science, Institute for Advanced Computer Studies, and Joint Center for Quantum Information and Computer Science, University of Maryland

While recent work suggests that quantum computers can speed up the solution of semidefinite programs, little is known about the quantum complexity of more general convex optimization. We present a quantum algorithm that can optimize a convex function over an n-dimensional convex body using $\tilde{O}(n)$ queries to oracles that evaluate the objective function and determine membership in the convex body. This represents a quadratic improvement over the best-known classical algorithm. We also study limitations on the power of quantum computers for general convex optimization, showing that it requires $\tilde{\Omega}(\sqrt{n})$ evaluation queries and $\Omega(\sqrt{n})$ membership queries.

1 Introduction

Convex optimization has been a central topic in the study of mathematical optimization, theoretical computer science, and operations research over the last several decades. On the one hand, it has been used to develop numerous algorithmic techniques for problems in combinatorial optimization, machine learning, signal processing, and other areas. On the other hand, it is a major class of optimization problems that admits efficient classical algorithms [5, 12]. Approaches to convex optimization include the ellipsoid method [12], interior-point methods [10, 17], cutting-plane methods [18, 28], and random walks [16, 23].

The fastest known classical algorithm for general convex optimization solves an instance of dimension n using $\tilde{O}(n^2)$ queries to oracles for the convex body and the objective function, and runs in time $\tilde{O}(n^3)$ [21].[1] The novel step of [21] is a construction of a separation oracle by a subgradient calculation with $O(n)$ objective function calls and $O(n)$ extra time. It then relies on a reduction from optimization to separation that makes $\tilde{O}(n)$ separation oracle calls and runs in time $\tilde{O}(n^3)$ [22]. Although it is unclear whether the query complexity of $\tilde{O}(n^2)$ is optimal for all possible classical algorithms, it is the best possible result using the above framework. This is because it takes $\tilde{\Omega}(n)$ queries to compute the (sub)gradient (see Section A.1) and it also requires $\Omega(n)$ queries to produce an optimization oracle from a separation oracle (see [25] and [24, Section 10.2.2]).

It is natural to ask whether quantum computers can solve convex optimization problems faster. Recently, there has been significant progress on quantum algorithms for solving a

[1] The notation $\tilde{O}$ suppresses poly-logarithmic factors in n, R, r, ϵ, i.e., $\tilde{O}(f(n)) = f(n)\log^{O(1)}(nR/r\epsilon)$.

special class of convex optimization problems called semidefinite programs (SDPs). SDPs generalize the better-known linear programs (LPs) by allowing positive semidefinite matrices as variables. For an SDP with n-dimensional, s-sparse input matrices and m constraints, the best known classical algorithm [22] finds a solution in time $\tilde{O}(m(m^2+n^\omega+mns)\operatorname{poly}\log(1/\epsilon))$, where ω is the exponent of matrix multiplication and ϵ is the accuracy of the solution. Brandão and Svore gave the first quantum algorithm for SDPs with worst-case running time $\tilde{O}(\sqrt{mn}s^2(Rr/\varepsilon)^{32})$, where R and r upper bound the norms of the optimal primal and dual solutions, respectively [7]. Compared to the aforementioned classical SDP solver [22], this gives a polynomial speedup in m and n. Van Apeldoorn et al. [3] further improved the running time of a quantum SDP solver to $\tilde{O}(\sqrt{mn}s^2(Rr/\epsilon)^8)$, which was subsequently improved to $\tilde{O}((\sqrt{m}+\sqrt{n}(Rr/\epsilon))s(Rr/\epsilon)^4)$ [2, 6]. The latter result is tight in the dependence of m and n since there is a quantum lower bound of $\Omega(\sqrt{m}+\sqrt{n})$ for constant R, r, s, ϵ [7].

However, semidefinite programming is a structured form of convex optimization that does not capture the problem in general. In particular, SDPs are specified by positive semidefinite matrices, and their solution is related to well-understood tasks in quantum computation such as solving linear systems (e.g., [9, 13]) and Gibbs sampling (e.g., [2, 6]). General convex optimization need not include such structural information, instead only offering the promise that the objective function and constraints are convex. Currently, little is known about whether quantum computers could provide speedups for general convex optimization. Our goal is to shed light on this question.

1.1 Convex optimization

We consider the following general minimization problem:

$$\min_{x\in K} f(x), \text{ where } K \subseteq \mathbb{R}^n \text{ is a convex set and } f\colon K \to \mathbb{R} \text{ is a convex function.} \tag{1.1}$$

We assume we are given upper and lower bounds on the function values, namely $m \leq \min_{x\in K} f(x) \leq M$, and inner and outer bounds on the convex set K, namely

$$B_2(0,r) \subseteq K \subseteq B_2(0,R), \tag{1.2}$$

where $B_2(x,l)$ is the ball of radius l in L_2 norm centered at $x \in \mathbb{R}^n$. We ask for a solution $\tilde{x} \in K$ with precision ϵ, in the sense that

$$f(\tilde{x}) \leq \min_{x\in K} f(x) + \epsilon. \tag{1.3}$$

We consider the very general setting where the convex body K and convex function f are only specified by oracles. In particular, we have:

- A *membership oracle* O_K for K, which determines whether a given $x \in \mathbb{R}^n$ belongs to K;
- An *evaluation oracle* O_f for f, which outputs $f(x)$ for a given $x \in K$.

Convex optimization has been well-studied in the model of membership and evaluation oracles since this provides a reasonable level of abstraction of K and f, and it helps illuminate the algorithmic relationship between the optimization problem and the relatively simpler task

of determining membership [12, 21, 22]. The efficiency of convex optimization is then measured by the number of queries to the oracles (i.e., the *query complexity*) and the total number of other elementary gates (i.e., the *gate complexity*).

It is well known that a general bounded convex optimization problem is equivalent to one with a linear objective function over a different bounded convex set. In particular, if promised that $\min_{x \in K} f(x) \leq M$, (1.1) is equivalent to the problem

$$\min_{x' \in \mathbb{R},\, x \in K} x' \quad \text{such that} \quad f(x) \leq x' \leq M. \tag{1.4}$$

Observe that a membership query to the new convex set

$$K' := \{(x', x) \in \mathbb{R} \times K \mid f(x) \leq x' \leq M\} \tag{1.5}$$

can be implemented with one query to the membership oracle for K and one query to the evaluation oracle for f. Thus the ability to optimize a linear function

$$\min_{x \in K} c^T x \tag{1.6}$$

for any $c \in \mathbb{R}^n$ and convex set $K \subseteq \mathbb{R}^n$ is essentially equivalent to solving a general convex optimization problem. A procedure to solve such a problem for any specified c is known as an *optimization oracle*. Thus convex optimization reduces to implementing optimization oracles over general convex sets (Lemma 2.1). The related concept of a *separation oracle* takes as input a point $p \notin K$ and outputs a hyperplane separating p from K.

In the quantum setting, we model oracles by unitary operators instead of classical procedures. In particular, in the quantum model of membership and evaluation oracles, we are promised to have unitaries O_K and O_f such that

- For any $x \in \mathbb{R}^n$, $O_K|x, 0\rangle = |x, \delta[x \in K]\rangle$, where $\delta[P]$ is 1 if P is true and 0 if P is false;
- For any $x \in \mathbb{R}^n$, $O_f|x, 0\rangle = |x, f(x)\rangle$.

In other words, we allow coherent superpositions of queries to both oracles. If the classical oracles can be implemented by explicit circuits, then the corresponding quantum oracles can be implemented by quantum circuits of about the same size, so the quantum query model provides a useful framework for understanding the quantum complexity of convex optimization.

1.2 Contributions

We now describe the main contributions of this paper. Our first main result is a quantum algorithm for optimizing a convex function over a convex body. Specifically, we show the following:

Theorem 1.1. *There is a quantum algorithm for minimizing a convex function f over a convex set $K \subseteq \mathbb{R}^n$ using $\tilde{O}(n)$ queries to an evaluation oracle for f and $\tilde{O}(n)$ queries to a membership oracle for K. The gate complexity of this algorithm is $\tilde{O}(n^3)$.*

Recall that the state-of-the-art classical algorithm [21] for general convex optimization with evaluation and membership oracles uses $\tilde{O}(n^2)$ queries to each. Thus our algorithm provides a quadratic improvement over the best known classical result. While the query

complexity of [21] is not known to be tight, it is the best possible result that can be achieved using subgradient computation to implement a separation oracle, as discussed above.

The proof of Theorem 1.1 follows the aforementioned classical strategy of constructing a separating hyperplane for any given point outside the convex body [21]. We find this hyperplane using a fast quantum algorithm for gradient estimation using $\tilde{O}(1)$ evaluation queries,[2] as first proposed by Jordan [15] and later refined by [11] with more rigorous analysis. However, finding a suitable hyperplane in general requires calculating approximate subgradients of convex functions that may not be differentiable, whereas the algorithms in [15] and [11] both require bounded second derivatives or more stringent conditions. To address this issue, we introduce classical randomness into the algorithm to produce a suitable approximate subgradient with $\tilde{O}(1)$ evaluation queries, and show how to use such an approximate subgradient in the separation framework to produce a faster quantum algorithm.

Our new quantum algorithm for subgradient computation is the source of the quantum speedup of the entire algorithm and establishes a separation in query complexity for the subgradient computation between quantum ($\tilde{O}(1)$) and classical ($\tilde{\Omega}(n)$, see Section A.1) algorithms. This subroutine could also be of independent interest, in particular in the study of quantum algorithms based on gradient descent and its variants (e.g., [19, 27]).

Our techniques for finding an approximate subgradient only require an approximate oracle for the function to be differentiated. Theorem 1.1 also applies if the membership and evaluation oracles are given with error that is polynomially related to the required precision in minimizing the convex function (see Theorem 2.6). Precise definitions for these oracles with error can be found in Section 2.1.

On the other hand, we also aim to establish corresponding quantum lower bounds to understand the potential for quantum speedups for convex optimization. To this end, we prove:

Theorem 1.2. *There exists a convex body $K \subseteq \mathbb{R}^n$, a convex function f on K, and a precision $\epsilon > 0$, such that a quantum algorithm needs at least $\Omega(\sqrt{n})$ queries to a membership oracle for K and $\Omega(\sqrt{n}/\log n)$ queries to an evaluation oracle for f to output a point $\tilde{x}$ satisfying*

$$f(\tilde{x}) \leq \min_{x \in K} f(x) + \epsilon \tag{1.7}$$

with high success probability (say, at least 0.8).

We establish the query lower bound on the membership oracle by reductions from search with wildcards [1]. The lower bound on evaluation queries uses a similar reduction, but this only works for an evaluation oracle with low precision. To prove a lower bound on precise evaluation queries, we propose a discretization technique that relates the difficulty of the continuous problem to a corresponding discrete one. This approach might be of independent interest since optimization problems naturally have continuous inputs and outputs, whereas most previous work on quantum lower bounds focuses on discrete inputs. Using this technique, we can simulate one perfectly precise query by one low-precision query at discretized points, thereby establishing the evaluation lower bound as claimed in Theorem 1.2. As a side point, this evaluation lower bound holds even for an unconstrained convex optimization problem

[2] Here $\tilde{O}(1)$ has the same definition as footnote 1, i.e., $\tilde{O}(1) = \log^{O(1)}(nR/r\epsilon)$.

on $\mathbb{R}^n$, which might be of independent interest since this setting has also been well-studied classically [5, 24–26].

We summarize our main results in Table 1.

	Classical bounds	Quantum bounds (this paper)
Membership queries	$\tilde{O}(n^2)$ [21], $\Omega(n)$ [20]	$\tilde{O}(n)$, $\Omega(\sqrt{n})$
Evaluation queries	$\tilde{O}(n^2)$ [21], $\Omega(n)$ [20]	$\tilde{O}(n)$, $\tilde{\Omega}(\sqrt{n})$
Time complexity	$\tilde{O}(n^3)$ [21]	$\tilde{O}(n^3)$

Table 1: Summary of classical and quantum complexities of convex optimization.

1.3 Overview of techniques

1.3.1 Upper bound

To prove our upper bound result in Theorem 1.1, we use the well-known reduction from general convex optimization to the case of a linear objective function, which simplifies the problem to implementing an optimization oracle using queries to a membership oracle (Lemma 2.1). For the reduction from optimization to membership, we follow the best known classical result in [21] which implements an optimization oracle using $\tilde{O}(n^2)$ membership queries and $\tilde{O}(n^3)$ arithmetic operations. In [21], the authors first show a reduction from separation oracles to membership oracles that uses $\tilde{O}(n)$ queries and then use a result from [22] to implement an optimization oracle using $\tilde{O}(n)$ queries to a separation oracle, giving an overall query complexity of $\tilde{O}(n^2)$.

The reduction from separation to membership involves the calculation of a *height function* defined by the authors (see Eq. (2.34)), whose evaluation oracle can be implemented in terms of the membership oracle of the original set. A separating hyperplane is determined by computing a subgradient, which already takes $\tilde{O}(n)$ queries. In fact, it is not hard to see that any classical algorithm requires $\tilde{\Omega}(n)$ classical queries (see Section A.1), so this part of the algorithm cannot be improved classically. The possibility of using the quantum Fourier transform to compute the gradient of a function using $\tilde{O}(1)$ evaluation queries ([11, 15]) suggests the possibility of replacing the subgradient procedure with a faster quantum algorithm. However, the techniques described in [11, 15] require the function in question to have bounded second (or even higher) derivatives, and the height function is only guaranteed to be Lipschitz continuous (Definition 2.9) and in general is not even differentiable.

To compute subgradients of general (non-differentiable) convex functions, we introduce classical randomness (taking inspiration from [21]) and construct a quantum subgradient algorithm that uses $\tilde{O}(1)$ queries. Our proof of correctness (Section 2.2) has three main steps:

1. We analyze the average error incurred when computing the gradient using the quantum Fourier transform. Specifically, we show that this approach succeeds if the function has bounded second derivatives in the vicinity of the point where the gradient is to be calculated (see Algorithm 1, Algorithm 2, and Lemma 2.3). Some of our calculations are inspired by [11].

2. We use the technique of *mollifier functions* (a common tool in functional analysis [14], suggested to us by [20] in the context of [21]) to show that it is sufficient to treat in-

finitely differentiable functions (the mollified functions) with bounded first derivatives (but possibly large second derivatives). In particular, it is sufficient to output an approximate gradient of the mollified function at a point near the original point where the subgradient is to be calculated (see Lemma 2.4).

3. We prove that convex functions with bounded first derivatives have second derivatives that lie below a certain threshold with high probability for a random point in the vicinity of the original point (Lemma 2.5). Furthermore, we show that a bound on the second derivatives can be chosen so that the smooth gradient calculation techniques work on a sufficiently large fraction of the neighborhood of the original point, ensuring that the final subgradient error is small (see Algorithm 3 and Theorem 2.2).

The new quantum subgradient algorithm is then used to construct a separation oracle as in [21] (and a similar calculation is carried out in Theorem 2.3). Finally the reduction from [22] is used to construct the optimization oracle using $\tilde{O}(n)$ separation queries. From Lemma 2.1, this shows that the general convex optimization problem can be solved using $\tilde{O}(n)$ membership and evaluation queries and $\tilde{O}(n^3)$ gates.

1.3.2 Lower bound

We prove our quantum lower bounds on membership and evaluation queries separately before showing how to combine them into a single optimization problem. Both lower bounds work over n-dimensional hypercubes.

In particular, we prove both lower bounds by reductions from search with wildcards [1]. In this problem, we are given an n-bit binary string s and the task is to determine all bits of s using wildcard queries that check the correctness of any subset of the bits of s: more formally, the input in the wildcard model is a pair (T, y) where $T \subseteq [n]$ and $y \in \{0,1\}^{|T|}$, and the query returns 1 if $s_{|T} = y$ (here the notation $s_{|T}$ represents the subset of the bits of s restricted to T). Reference [1] shows that the quantum query complexity of search with wildcards is $\Omega(\sqrt{n})$.

For our lower bound on membership queries, we consider a simple objective function, the sum of all coordinates $\sum_{i=1}^{n} x_i$. In other words, we take $c = \mathbf{1}^n$ in (1.6). However, the position of the hypercube is unknown, and to solve the optimization problem (formally stated in Definition 3.1), one must use the membership oracle to locate it.

Specifically, the hypercube takes the form $\bigtimes_{i=1}^{n}[s_i - 2, s_i + 1]$ (where $\times$ is the Cartesian product) for some offset binary string $s \in \{0,1\}^n$. In Section 3.1, we prove:

- Any query $x \in \mathbb{R}^n$ to the membership oracle of this problem can be simulated by one query to the search-with-wildcards oracle for s. To achieve this, we divide the n coordinates of x into four sets: $T_{x,0}$ for those in $[-2,-1)$, $T_{x,1}$ for those in $(1,2]$, $T_{x,\text{mid}}$ for those in $[-1,1]$, and $T_{x,\text{out}}$ for the rest. Notice that $T_{x,\text{mid}}$ corresponds to the coordinates that are always in the hypercube and $T_{x,\text{out}}$ corresponds to the coordinates that are always out of the hypercube; $T_{x,0}$ (resp., $T_{x,1}$) includes the coordinates for which $s_i = 0$ (resp., $s_i = 1$) impacts the membership in the hypercube. We prove in Section 3.1 that a wildcard query with $T = T_{x,0} \cup T_{x,1}$ can simulate a membership query to x.

- The solution of the sum-of-coordinates optimization problem explicitly gives s, i.e., it solves search with wildcards. This is because this solution must be close to the point $(s_1 - 2, \ldots, s_n - 2)$, and applying integer rounding would recover s.

These two points establish the reduction of search with wildcards to the optimization problem, and hence establishes the $\Omega(\sqrt{n})$ membership quantum lower bound in Theorem 1.2 (see Theorem 3.2).

For our lower bound on evaluation queries, we assume that membership is trivial by fixing the hypercube at $\mathcal{C} = [0,1]^n$. We then consider optimizing the max-norm function

$$f(x) = \max_{i \in [n]} |x_i - c_i| \tag{1.8}$$

for some unknown $c \in \{0,1\}^n$. Notice that learning c is equivalent to solving the optimization problem; in particular, outputting an $\tilde{x} \in \mathcal{C}$ satisfying (1.3) with $\epsilon = 1/3$ would determine the string c. This follows because for all $i \in [n]$, we have $|\tilde{x}_i - c_i| \leq \max_{i \in [n]} |\tilde{x}_i - c_i| \leq 1/3$, and c_i must be the integer rounding of $\tilde{x}_i$, i.e., $c_i = 0$ if $\tilde{x}_i \in [0, 1/2)$ and $c_i = 1$ if $\tilde{x}_i \in [1/2, 1]$. On the other hand, if we know c, then we know the optimum $x = c$.

We prove an $\Omega(\sqrt{n}/\log n)$ lower bound on evaluation queries for learning c. Our proof, which appears in Section 3.2, is composed of three steps:

1) We first prove a weaker lower bound with respect to the precision of the evaluation oracle. Specifically, if $f(x)$ is specified with b bits of precision, then using binary search, a query to $f(x)$ can be simulated by b queries to an oracle that inputs $(f(x), t)$ for some $t \in \mathbb{R}$ and returns 1 if $f(x) \leq t$ and returns 0 otherwise. We further without loss of generality assume $x \in [0,1]^n$. If $x \notin [0,1]^n$, we assign a penalty of the L_1 distance between x and its projection $\pi(x)$ onto $[0,1]^n$; by doing so, $f(\pi(x))$ and x fully characterizes $f(x)$ (see (3.18)). Therefore, $f(x) \in [0,1]$, and $f(x)$ having b bits of precision is equivalent to having precision 2^{-b}.

   Similar to the interval dividing strategy in the proof of the membership lower bound, we prove that one query to such an oracle can be simulated by one query to the search-with-wildcards oracle for s. Furthermore, the solution of the max-norm optimization problem explicitly gives s, i.e., it solves the search-with-wildcards problem. This establishes the reduction to search with wildcards, and hence establishes an $\Omega(\sqrt{n}/b)$ lower bound on the number of quantum queries to the evaluation oracle f with precision 2^{-b} (see Lemma 3.1).

2) Next, we introduce a technique we call *discretization*, which effectively simulates queries over an (uncountably) infinite set by queries over a discrete set. This technique might be of independent interest since proving lower bounds on functions with an infinite domain can be challenging.

   We observe that the problem of optimizing (1.8) has the following property: if we are given two strings $x, x' \in [0,1]^n$ such that $x_1, \ldots, x_n, 1-x_1, \ldots, 1-x_n$ and $x'_1, \ldots, x'_n, 1-x'_1, \ldots, 1-x'_n$ have the same ordering (for instance, strings $x = (0.1, 0.2, 0.7)$ and $x' = (0.1, 0.3, 0.6)$ both have the ordering $x_1 \leq x_2 \leq 1 - x_3 \leq x_3 \leq 1 - x_2 \leq 1 - x_1$), then

$$\arg\max_{i \in [n]} |x_i - c_i| = \arg\max_{i \in [n]} |x'_i - c_i|. \tag{1.9}$$

Furthermore, if $x'_1, \ldots, x'_n, 1-x'_1, \ldots, 1-x'_n$ are $2n$ different numbers, then knowing the value of $f(x')$ implies the value of the arg max in (1.9) (denoted i^*) and the corresponding c_{i^*}, and we can subsequently recover $f(x)$ given x since $f(x) = |x_{i^*} - c_{i^*}|$. In other words, $f(x)$ can be computed given x and $f(x')$.

Therefore, it suffices to consider all possible ways of ordering $2n$ numbers, rendering the problem discrete. Without loss of generality, we focus on x' satisfying $\{x'_1, \ldots, x'_n, 1 - x'_1, \ldots, 1 - x'_n\} = \{\frac{1}{2n+1}, \ldots, \frac{2n}{2n+1}\}$, and we denote the set of all such x' by D_n (see also (3.34)). In Lemma 3.4, we prove that one classical (resp., quantum) evaluation query from $[0,1]^n$ can be simulated by one classical evaluation query (resp., two quantum evaluation queries) from D_n using Algorithm 5. To illustrate this, we give a concrete example with $n = 3$ in Section 3.2.2.

3) Finally, we use discretization to show that one perfectly precise query to f can be simulated by one query to f with precision $\frac{1}{5n}$; in other words, b in step 1) is at most $\lceil \log_2 5n \rceil = O(\log n)$ (see Lemma 3.3). This is because by discretization, the input domain can be limited to the discrete set D_n. Notice that for any $x \in D_n$, $f(x)$ is an integer multiple of $\frac{1}{2n+1}$; even if $f(x)$ can only be computed with precision $\frac{1}{5n}$, we can round it to the closest integer multiple of $\frac{1}{2n+1}$ which is exactly $f(x)$, since the distance $\frac{2n+1}{5n} < \frac{1}{2}$. As a result, we can precisely compute $f(x)$ for all $x \in D_n$, and thus by discretization we can precisely compute $f(x)$ for all $x \in [0,1]^n$.

In all, the three steps above establish an $\Omega(\sqrt{n}/\log n)$ quantum lower bound on evaluation queries to solve the problem in Eq. (1.8) (see Theorem 3.2). In particular, this lower bound is proved for an unconstrained convex optimization problem on $\mathbb{R}^n$, which might be of independent interest.

As a side result, we prove that our quantum lower bound is optimal for the problem in (1.8) (up to poly-logarithmic factors in n), as we can prove a matching $\tilde{O}(\sqrt{n})$ upper bound (Theorem C.1). Therefore, a better quantum lower bound on the number of evaluation queries for convex optimization would require studying an essentially different problem.

Having established lower bounds on both membership and evaluation queries, we combine them to give Theorem 1.2. This is achieved by considering an optimization problem of dimension $2n$; the first n coordinates compose the sum-of-coordinates function in Section 3.1, and the last n coordinates compose the max-norm function in Section 3.2. We then concatenate both parts and prove Theorem 1.2 via reductions to the membership and evaluation lower bounds, respectively (see Section 3.3).

In addition, all lower bounds described above can be adapted to a convex body that is contained in the unit hypercube and that contains the discrete set D_n to facilitate discretization; we present a "smoothed" hypercube (see Section 3.4) as a specific example.

1.4 Open questions

This work leaves several natural open questions for future investigation. In particular:

- Can we close the gap for both membership and evaluation queries? Our upper bounds on both oracles in Theorem 1.1 uses $\tilde{O}(n)$ queries, whereas the lower bounds of Theorem 1.2 are only $\tilde{\Omega}(\sqrt{n})$.

- Can we improve the time complexity of our quantum algorithm? The time complexity $\tilde{O}(n^3)$ of our current quantum algorithm matches that of the classical state-of-the-art algorithm [21] since our second step, the reduction from optimization to separation, is entirely classical. Is it possible to improve this reduction quantumly?

- What is the quantum complexity of convex optimization with a first-order oracle (i.e., with direct access to the gradient of the objective function)? This model has been widely considered in the classical literature (see for example Ref. [26]).

Organization. Our quantum upper bounds are given in Section 2 and lower bounds are given in Section 3. Appendices present auxiliary lemmas (Section A) and proof details for upper bounds (Section B) and lower bounds (Section C), respectively.

Related independent work. In independent simultaneous work, van Apeldoorn, Gilyén, Gribling, and de Wolf [4] establish a similar upper bound, showing that $\tilde{O}(n)$ quantum queries to a membership oracle suffice to optimize a linear function over a convex body (i.e., to implement an optimization oracle). Their proof follows a similar strategy to ours, using a quantum algorithm for evaluating gradients in $\tilde{O}(1)$ queries to implement a separation oracle. As in our approach, they use a randomly sampled point in the neighborhood of the point where the subgradient is to be calculated. The only major difference is that they use finite approximations of the gradient and second derivatives, whereas we use these quantities in their original form and give an argument based on mollifier functions to ensure that they are well defined.

Reference [4] also establishes quantum lower bounds on the query complexity of convex optimization, showing in particular that $\Omega(\sqrt{n})$ quantum queries to a separation oracle are needed to implement an optimization oracle, implying an $\Omega(\sqrt{n})$ quantum lower bound on the number of membership queries required to optimize a convex function. While Ref. [4] does not explicitly focus on evaluation queries, those authors have pointed out to us that an $\Omega(\sqrt{n})$ lower bound on evaluation queries can be obtained from their lower bound on membership queries (although our approach gives a bound with a better Lipschitz parameter).

2 Upper bound

In this section, we prove:

Theorem 2.1. *An optimization oracle for a convex set $K \subseteq \mathbb{R}^n$ can be implemented using $\tilde{O}(n)$ quantum queries to a membership oracle for K, with gate complexity $\tilde{O}(n^3)$.*

The following lemma shows the equivalence of optimization oracles to a general convex optimization problem.

Lemma 2.1. *Suppose a reduction from an optimization oracle to a membership oracle for convex sets requires $O(g(n))$ queries to the membership oracle. Then the problem of optimizing a convex function over a convex set can be solved using $O(g(n))$ queries to both the membership oracle and the evaluation oracle.*

Proof. The problem $\min_{x \in K} f(x)$ reduces to the problem $\min_{(x',x) \in K'} x'$ where K' is defined as in (1.4). K' is the intersection of convex sets and is therefore itself convex. A membership oracle for K' can be implemented using 1 query each to the membership oracle for K and the evaluation oracle for f. Since $O(g(n))$ queries to the membership oracle for K' are sufficient to optimize any linear function, the result follows. □

Theorem 1.1 directly follows from Theorem 2.1 and Lemma 2.1.

Overview. This part of the paper is organized following the plan outlined in Section 1.3.1. Precise definitions of oracles and other relevant terminology appear in Section 2.1. Section 2.2 develops a fast quantum subgradient procedure that can be used in the classical reduction from optimization to membership. This is done in two parts:

1. Section 2.2.1 presents an algorithm based on the quantum Fourier transform that calculates the gradient of a function with bounded second derivatives (i.e., a β-smooth function) with bounded expected one-norm error.

2. Section 2.2.2 uses mollification to restrict the analysis to infinitely differentiable functions without loss of generality, and then uses classical randomness to eliminate the need for bounded second derivatives.

In Section 2.3 we show that the new quantum subgradient algorithm fits into the classical reduction from [21]. Finally, we describe the reduction from optimization to membership in Section 2.4.

2.1 Oracle definitions

In this section, we provide precise definitions for the oracles for convex sets and functions that we use in our algorithm and its analysis. We also provide precise definitions of Lipschitz continuity and β-smoothness, which we will require in the rest of the section.

Definition 2.1 (Ball in L_p norm)**.** *The ball of radius $r > 0$ in L_p norm $\|\cdot\|_p$ centered at $x \in \mathbb{R}^n$ is $B_p(x, r) := \{y \in \mathbb{R}_n \mid \|x - y\|_p \leq r\}$.*

Definition 2.2 (Interior of a convex set)**.** *For any $\delta > 0$, the δ-interior of a convex set K is defined as $B_2(K, -\delta) := \{x \mid B_2(x, \delta) \subseteq K\}$.*

Definition 2.3 (Neighborhood of a convex set)**.** *For any $\delta > 0$, the δ-neighborhood of a convex set K is defined as $B_2(K, \delta) := \{x \mid \exists\, y \in K \text{ s.t. } \|x - y\|_2 \leq \delta\}$.*

Definition 2.4 (Evaluation oracle)**.** *When queried with $x \in \mathbb{R}^n$ and $\delta > 0$, output α such that $|\alpha - f(x)| \leq \delta$. We use $\mathrm{EVAL}_\delta(f)$ to denote the time complexity. The classical procedure or quantum unitary representing the oracle is denoted by O_f.*

Definition 2.5 (Membership oracle)**.** *When queried with $x \in \mathbb{R}^n$ and $\delta > 0$, output an assertion that $x \in B_2(K, \delta)$ or $x \notin B_2(K, -\delta)$. The time complexity is denoted by $\mathrm{MEM}_\delta(K)$. The classical procedure or quantum unitary representing the membership oracle is denoted by O_K.*

Definition 2.6 (Separation oracle)**.** *When queried with $x \in \mathbb{R}^n$ and $\delta > 0$, with probability $1 - \delta$, either*

- *assert $x \in B_2(K, \delta)$ or*
- *output a unit vector $\hat{c}$ such that $\hat{c}^T x \leq \hat{c}^T y + \delta$ for all $y \in B_2(K, -\delta)$.*

The time complexity is denoted by $\mathrm{SEP}_\delta(K)$.

Definition 2.7 (Optimization oracle)**.** *When queried with a unit vector c, find $y \in \mathbb{R}^n$ such that $c^T x \leq c^T y + \delta$ for all $x \in B_2(K, -\delta)$ or asserts that $B_2(K, \delta)$ is empty. The time complexity of the oracle is denoted by $\mathrm{OPT}_\delta(K)$.*

Definition 2.8 (Subgradient)**.** *A subgradient of a convex function $f\colon \mathbb{R}^n \to \mathbb{R}$ at x, is a vector g such that*

$$f(y) \geq f(x) + \langle g, y - x \rangle \tag{2.1}$$

for all $y \in \mathbb{R}^n$. For a differentiable convex function, the gradient is the only subgradient. The set of subgradients of f at x is called the subdifferential at x and denoted by $\partial f(x)$.

Definition 2.9 (L-Lipschitz continuity)**.** *A function $f\colon \mathbb{R}^n \to \mathbb{R}$ is said to be L-Lipschitz continuous (or simply L-Lipschitz) in a set S if for all $x \in S$, $\|g\|_\infty \leq L$ for any $g \in \partial f(x)$. An immediate consequence of this is that for any $x, y \in S$,*

$$|f(y) - f(x)| \leq L\|y - x\|_\infty. \tag{2.2}$$

Definition 2.10 (β-smoothness)**.** *A function $f\colon \mathbb{R}^n \to \mathbb{R}$ is said to be β-smooth in a set S if for all $x \in S$, the magnitudes of the second derivatives of f in all directions are bounded by β. This also means that the largest magnitude of an eigenvalue of the Hessian $\nabla^2 f(x)$ is at most β. Consequently, for any $x, y \in S$, we have*

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\beta}{2}\|y - x\|_\infty^2. \tag{2.3}$$

2.2 Evaluation to subgradient

In this section we present a procedure that, given an evaluation oracle for an L-Lipschitz continuous function $f\colon \mathbb{R}^n \to \mathbb{R}$ with evaluation error at most $\epsilon > 0$, a point $x \in \mathbb{R}^n$, and an "approximation scale" factor $r_1 > 0$, computes an approximate subgradient $\tilde{g}$ of f at x. Specifically, $\tilde{g}$ satisfies

$$f(q) \geq f(x) + \langle \tilde{g}, q - x \rangle - \zeta\|q - x\|_\infty - 4nr_1 L \tag{2.4}$$

for all $q \in \mathbb{R}^n$, where $\mathbb{E}\zeta \leq \xi(r_1, \epsilon)$ and ξ must monotonically increase with ϵ as ϵ^α for some $\alpha > 0$. Here ζ is the error in the subgradient that is bounded in expectation by the function ξ.

2.2.1 Smooth functions

We first describe how to approximate the gradient of a smooth function. Algorithm 1 and Algorithm 2 use techniques from [15] and [11] to evaluate the gradient of a function with bounded second derivatives in the neighborhood of the evaluation point. To analyze their behavior, we begin with the following lemma showing that Algorithm 1 provides a good estimate of the gradient with bounded failure probability.

Algorithm 1: `GradientEstimate`$(f, \epsilon, L, \beta, x_0)$

Data: Function f, evaluation error ϵ, Lipschitz constant L, smoothness parameter β, and point x_0.

Define

- $l = 2\sqrt{\epsilon/n\beta}$ to be the size of the grid used,
- $b \in \mathbb{N}$ such that $\frac{24\pi\sqrt{n\epsilon\beta}}{L} \leq \frac{1}{2^b} = \frac{1}{N} \leq \frac{48\pi\sqrt{n\epsilon\beta}}{L}$,
- $b_0 \in \mathbb{N}$ such that $\frac{N\epsilon}{2Ll} \leq \frac{1}{2^{b_0}} = \frac{1}{N_0} \leq \frac{N\epsilon}{Ll}$,
- $F(x) = \frac{N}{2Ll}[f(x_0 + \frac{l}{N}(x - N/2)) - f(x_0)]$, and,
- $\gamma : \{0, 1, \ldots, N-1\} \to G := \{-N/2, -N/2+1, \ldots, N/2-1\}$ s.t. $\gamma(x) = x - N/2$.

Let O_F denote a unitary operation acting as $O_F |x\rangle = e^{2\pi i \tilde{F}(x)} |x\rangle$, where $|\tilde{F}(x) - F(x)| \leq \frac{1}{N_0}$, with x represented using b bits and $\tilde{F}(x)$ represented using b_0 bits.

1 Start with n b-bit registers set to 0 and Hadamard transform each to obtain

$$\frac{1}{\sqrt{N^n}} \sum_{x_1,\ldots,x_n \in \{0,1,\ldots,N-1\}} |x_1, \ldots, x_n\rangle ; \tag{2.5}$$

2 Perform the operation O_F and the map $|x\rangle \mapsto |\gamma(x)\rangle$ to obtain

$$\frac{1}{N^{n/2}} \sum_{g \in G^n} e^{2\pi i \tilde{F}(g)} |g\rangle ; \tag{2.6}$$

3 Apply the inverse QFT over G to each of the registers;

4 Measure the final state to get $k_1, k_2, \ldots, k_n$ and report $\tilde{g} = \frac{2L}{N}(k_1, k_2, \ldots, k_n)$ as the result.

Lemma 2.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz function that is specified by an evaluation oracle with error at most ϵ. Let f be β-smooth in $B_\infty(x, 2\sqrt{\epsilon/\beta})$, and let $\tilde{g}$ be the output of* `GradientEstimate`$(f, \epsilon, L, \beta, x_0)$ *(from Algorithm 1). Then*

$$\Pr\left[|\tilde{g}_i - \nabla f(x)_i| > 1500\sqrt{n\epsilon\beta}\right] < \frac{1}{3}, \quad \forall\, i \in [n]. \tag{2.7}$$

The proof of Lemma 2.2 is deferred to Lemma B.2 in the appendix.

Next we analyze Algorithm 2, which uses several calls to Algorithm 1 to provide an estimate of the gradient that is close in expected L_1 distance to the true value.

Lemma 2.3. *Let f be a convex, L-Lipshcitz continuous function that is specified by an evaluation oracle with error at most ϵ. Suppose f is β-smooth in $B_\infty(x, 2\sqrt{\epsilon/\beta})$. Let*

$$\tilde{g} = \texttt{SmoothQuantumGradient}(f, \epsilon, L, \beta, x) \tag{2.8}$$

(from Algorithm 2). Then for any $i \in [n]$, we have $|\tilde{g}_i| \leq L$ and $\mathbb{E}|\tilde{g}_i - \nabla f(x)_i| \leq 3000\sqrt{n\epsilon\beta}$; hence

$$\mathbb{E}\|\tilde{g} - \nabla f(x)\|_1 \leq 3000 n^{3/2}\sqrt{\epsilon\beta}. \tag{2.9}$$

If L, $1/\beta$, and $1/\epsilon$ are poly(n)*, the* `SmoothQuantumGradient` *algorithm uses $\tilde{O}(1)$ queries to the evaluation oracle and $\tilde{O}(n)$ gates.*

Algorithm 2: `SmoothQuantumGradient`$(f, \epsilon, L, \beta, x)$

Data: Function f, evaluation error ϵ, Lipschitz constant L, smoothness parameter β, and point x.

Set T such that $2e^{-T^2/24} \leq 750\sqrt{n\epsilon\beta}/L$;

for $t = 1, 2, \ldots, T$ **do**

  $e^{(t)} \leftarrow$ `GradientEstimate`$(f, \epsilon, L, \beta, x)$;

for $i = 1, 2, \ldots, n$ **do**

  If more than $T/2$ of $e_i^{(t)}$ lie in an interval of size $3000\sqrt{n\epsilon\beta}$, set $\tilde{g}_i$ to be the median of the points in that interval;

  Otherwise, set $\tilde{g}_i = 0$;

Output $\tilde{g}$.

Proof. For each dimension $i \in [n]$ and each iteration $t \in [T]$, consider the random variable

$$X_i^t = \begin{cases} 1 & \text{if } |e_i^{(t)} - \nabla f(x)_i| > 1500\sqrt{n\epsilon\beta} \\ 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

From the conditions on the function f, Lemma 2.2 applies to `GradientEstimate`$(f, \epsilon, L, \beta, x)$, and thus $\Pr(X_i^t = 1) < 1/3$. Thus, by the Chernoff bound, $\Pr\left[|\tilde{g}_i - \nabla f(x)_i| \leq 1500\sqrt{n\epsilon\beta}\right] > 1 - 2e^{-T^2/24} \geq 1 - 750\sqrt{n\epsilon\beta}/L$. In the remaining cases, $|\tilde{g}_i - \nabla f(x)_i| \leq 2L$ (see Line 4 of Algorithm 1). Thus $\mathbb{E}|\tilde{g}_i - \nabla f(x)_i| \leq 3000\sqrt{n\epsilon\beta}$ for all $i \in [n]$, and (2.9) follows.

The algorithm makes $T = \mathrm{poly}(\log(1/n\epsilon\beta))$ calls to a procedure that makes one query to the evaluation oracle. Thus the query complexity is $\tilde{O}(1)$. To evaluate the gate complexity, observe that we iterate over n dimensions, using $\mathrm{poly}(b) = \mathrm{poly}(\log(1/n\epsilon\beta))$ gates for the quantum Fourier transform over each. This process is repeated $T = \mathrm{poly}(\log(1/n\epsilon\beta))$ times. Thus the entire algorithm uses $\tilde{O}(n)$ gates. □

2.2.2 Extension to non-smooth functions

Now consider a general L-Lipschitz continuous convex function f. We show that any such function is close to a smooth function, and we consider the relationship between the subgradients of the original function and the gradient of its smooth approximation.

For any $\delta > 0$, let $m_\delta \colon \mathbb{R}^n \to \mathbb{R}$ be the *mollifier function of width* δ, defined as

$$m_\delta(x) := \begin{cases} \frac{1}{I_n} \exp\left(-\frac{1}{1-\|x/\delta\|_2^2}\right) & x \in B_2(0, \delta) \\ 0 & \text{otherwise,} \end{cases} \tag{2.11}$$

where I_n is chosen such that $\int_{B_2(0,\delta)} m_\delta(x)\, \mathrm{d}^n x = 1$. The mollification of f, denoted $F_\delta := f * m_\delta$, is obtained by convolving it with the mollifier function, i.e.,

$$F_\delta(x) = (f * m_\delta)(x) = \int_{\mathbb{R}^n} f(x - y) m_\delta(y)\, \mathrm{d}^n x. \tag{2.12}$$

The mollification of f has several key properties, as follows:

Proposition 2.1. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz convex function with mollification F_δ. Then*

(i) *F_δ is infinitely differentiable,*
(ii) *F_δ is convex,*
(iii) *F_δ is L-Lipschitz continuous, and*
(iv) *$|F_\delta(x) - f(x)| \le L\delta$.*

These properties of the mollifier function are well known in functional analysis [14]. For completeness a proof is provided in Lemma A.2.

Furthermore, an approximate gradient of the mollified function gives an approximate subgradient of the original function, as quantified by the following lemma.

Lemma 2.4. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be an infinitely differentiable L-Lipschitz continuous convex function with mollification F_δ. Then any $\tilde{g}$ satisfying $\|\tilde{g} - \nabla F_\delta(y)\|_1 = \zeta$ for some $y \in B_\infty(x, r_1)$ satisfies*

$$f(q) \ge f(x) + \langle \tilde{g}, q - x\rangle - \zeta\|q - x\|_\infty - 4nr_1L - 2L\delta. \tag{2.13}$$

Here ζ is the error in the subgradient and δ is the parameter used in the mollifier function.

Proof. For all $q \in \mathbb{R}^n$, convexity of F_δ implies

$$\begin{aligned} F_\delta(q) &\ge F_\delta(y) + \langle \nabla F_\delta(y), q - y\rangle && (2.14)\\ &= F_\delta(x) + \langle \nabla F_\delta(y), q - x\rangle + \langle \nabla F_\delta(y), x - y\rangle + (F_\delta(y) - F_\delta(x)) && (2.15)\\ &\ge F_\delta(x) + \langle \nabla F_\delta(y), q - x\rangle - 4nr_1L && (2.16)\\ &\ge F_\delta(x) + \langle \tilde{g}, q - x\rangle - \zeta\|q - x\|_\infty - 4nr_1L, && (2.17)\end{aligned}$$

so (2.13) follows from Proposition 2.1(iv). □

Now consider δ such that $L\delta \ll \epsilon$. Then the evaluation oracle with error ϵ for f is also an evaluation oracle for F_δ with error $\epsilon + L\delta \approx \epsilon$. Thus the given evaluation oracle is also the evaluation oracle for an infinitely differentiable convex function with the same Lipschitz constant and almost the same error, allowing us to analyze infinitely differentiable functions without loss of generality (as long as we make no claim about the second derivatives). This idea is made precise in Theorem 2.2. (Note that the mollification of f is never computed or estimated by our algorithm; it is only a tool for analysis.)

Unfortunately, Lemma 2.3 cannot be directly used to calculate subgradients for F_δ as $\delta \to 0$. This is because there exist convex functions (such as $f(x) = |x|$) where if $|f(x) - g(x)| \le \delta$ and $g(x)$ is β-smooth, then $\beta\delta \ge c$ for some constant c (see Lemma A.3 in the appendix). Thus using the `SmoothQuantumGradient` algorithm at $x = 0$ will give us a one-norm error of $3000n^{3/2}\sqrt{\epsilon\beta} \ge 3000n^{3/2}\sqrt{c}$, which is independent of ϵ.

To fix this issue, we take inspiration from [21] and introduce classical randomness into the gradient evaluation. In particular, the following lemma shows that for a Lipschitz continuous function, if we sample at random from the neighborhood of any given point, the probability of having large second derivatives is small. Let $y \sim Y$ indicate that y is sampled uniformly at random from the set Y. Also, let $\lambda(x)$ be the largest eigenvalue of the Hessian matrix $\nabla^2 f(x)$ at x. Since the Hessian is positive semidefinite, we have $\lambda(x) \le \Delta f(x) := \mathrm{Tr}(\nabla^2 f(x))$. Thus the second derivatives of f are upper bounded by $\Delta f(x)$.

Let $\eta(y)$ denote the area element on the surface $\partial B_\infty(x, r_1)$, defined as

$$\eta(y)_i := \begin{cases} 1 & \text{if } y_i - x_i \geq r_1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.18}$$

We have

$$\mathbb{E}_{y \sim B_\infty(x,r_1)} \Delta f(y) = \frac{1}{(2r_1)^n} \int\limits_{B_\infty(x,r_1)} \Delta f(y) \, \mathrm{d}^n y \tag{2.19}$$

$$= \frac{1}{(2r_1)^n} \int\limits_{\partial B_\infty(x,r_1)} \langle \nabla f(y), \eta(y) \rangle \, \mathrm{d}^{n-1} y \tag{2.20}$$

$$\leq \frac{1}{(2r_1)^n} (2n)(2r_1)^{n-1} L = \frac{nL}{r_1} \tag{2.21}$$

where (2.20) comes from the divergence theorem (the integral of the divergence of a vector field over a set is equal to the integral of the vector field over the surface of the set). This indicates that while the second derivatives of a Lipschitz continuous function can be unbounded at individual points, its expected value for a point uniformly sampled in an extended region is bounded.

Now, consider a grid of side length l (aligned with the coordinate axes) embedded in $B_\infty(x, r_1)$. We denote this grid by $G_{B_\infty(x,r_1),l}$. For any $i \in [n]$ and a y sampled uniformly from $G_{B_\infty(x,r_1),l}$, the expectation of the integral of the second directional derivative in the i^{th} coordinate direction over a segment from y to the point $y + le_i$ is

$$\mathbb{E}_{y \sim G_{B_\infty(x,r_1),l}} \left[\int_{y_i}^{y_i + le_i} \frac{\mathrm{d}^2 f(z)}{\mathrm{d} z_i^2} \, \mathrm{d} z_i \right] = \mathbb{E}_{y \sim G_{B_\infty(x,r_1),l}} \left[\int_0^l \frac{\mathrm{d}^2 f(y + te_i)}{dt^2} \mathrm{d}t \right] \leq \frac{Ll}{r_1}. \tag{2.22}$$

To see this, note that there are $2r_1/l$ segments of length l (corresponding to different points y) inside $B_\infty(y, r_1)$. The total integral of the directional derivative over these segments is upper bounded by the change in the i^{th} component of the gradient, which is in turn bounded by $2L$ due to the Lipschitz property of f.

Let $\Delta \colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ defined by

$$\Delta(y, z) := \left| f(z) - f(y) - \langle \nabla f(y), z - y \rangle \right| \tag{2.23}$$

be a function that quantifies the deviation from linearity of f between y and z. We now show the following lemma that bounds this deviation in the neighborhood of a randomly sampled point (with high probability).

Lemma 2.5. *Let $f \colon \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz continuous, infinitely differentiable, convex function. Then for a point y chosen uniformly from $G_{B_\infty(x,r_1),l}$, and any $p \in \mathbb{R}$ such that $p \geq n$,*

$$\Delta(y, z) \leq \frac{pnl^2 L}{r_1}, \quad \forall \, z \in B_\infty(y, l) \tag{2.24}$$

with probability at least $1 - \frac{n}{p}$.

Proof. Note that $\Delta(y, z)$ is a convex function of z and must attain its maximum at one of the extremal points (vertices) of the hypercube $B_\infty(y, l)$, which are the 2^n points of the form

$$\{y + ls \mid s \in \{-1, 1\}^n\}. \tag{2.25}$$

This is because every point in the hypercube is a convex combination of the vertices, so having a higher function value at an internal point than at all the vertices would violate convexity.

Consider a path from y to a vertex of $B_\infty(y, l)$ consisting of n segments of length l aligned along the n coordinate axes. For example, the path could move a distance l along $\pm e_1, \pm e_2, \ldots, \pm e_n$ until the vertex is reached. Using Markov's inequality with (2.22), we have for every coordinate direction $i \in [n]$,

$$\Pr_{y \sim G_{B_\infty(x, r_1), l}} \left[\int_y^{y + le_i} \frac{\mathrm{d}^2 f(z)}{\mathrm{d}z_i^2} \, \mathrm{d}z > \frac{pLl}{r_1} \right] \leq \frac{1}{p}. \tag{2.26}$$

Thus with probability at least $1 - \frac{1}{p}$, the increase in the deviation from linearity along each segment, as quantified by the function Δ, is at most $\frac{pl^2 L}{r_1}$. Using the union bound, with probability at least $1 - \frac{n}{p}$, the total deviation from linearity along the path is at most $\frac{pnl^2 L}{r_1}$ as claimed. □

Lemma 2.5 shows that with high probability a sampled point in $B_\infty(x, r_1)$ has a deviation from linearity in its neighborhood which is the same as that for a function with smoothness parameter $\frac{2pL}{r_1}$. The analysis of the gradient estimation procedure (Lemma 2.3) uses the smoothness of the function only to bound its deviation from linearity. Thus, Algorithm 2 can be applied as if to a function with smoothness parameter $\frac{2pL}{r_1}$. This observation is applied to find an approximate subgradient in Algorithm 3.

Algorithm 3: `QuantumSubgradient`(f, ϵ, L, x, r_1)

Data: Function f, evaluation error ϵ, Lipschitz constant L, point $x \in \mathbb{R}^n$, length $r_1 > 0$.

1 Sample $y \sim G_{B_\infty(x, r_1), l}$;

2 Output $\tilde{g} =$ `SmoothQuantumGradient`$(f, \epsilon, L, 2n^{1/3} L / r_1^{2/3} \epsilon^{1/3}, y)$.

Theorem 2.2. *Let f be a convex, L-Lipschitz function that is specified by an evaluation oracle with error $\epsilon < \min\{1, r_1/n^2\}$. Let $\tilde{g} =$ `QuantumSubgradient`(f, ϵ, L, x, r_1) (from Algorithm 3). Then for all $q \in \mathbb{R}^n$,*

$$f(q) \geq f(x) + \langle \tilde{g}, q - x \rangle - \zeta \|q - x\|_\infty - 4nr_1 L, \tag{2.27}$$

where $\mathbb{E}\zeta \leq \frac{5000 L n^{5/3} \epsilon^{1/3}}{r_1^{1/3}}$.

Proof. Consider F_δ such that $L\delta \ll \epsilon$. From Proposition 2.1, F_δ is infinitely differentiable, convex, and L-Lipschitz. The given evaluation oracle for f is also an evaluation oracle for F_δ with error $\epsilon_1 = \epsilon + L\delta$.

Assume without loss of generality that $L \geq 1$ (if not, the algorithm can be run with $L = 1$). Set $p = r_1^{1/3} n^{1/3} / \epsilon_1^{1/3}$ ($n/p < 1$ by the assumption on ϵ). As observed above, Lemma 2.5 shows

that for $y \in G_{B_\infty(x,r_1),l}$, Algorithm 2 produces $g = \texttt{SmoothQuantumGradient}(F_\delta, \epsilon_1, L, \frac{2pL}{r_1}, y)$ correctly with probability at least $1 - n/p$. Thus for each $i \in [n]$, we have:

1. With probability at least $1 - n^{2/3}\epsilon_1^{1/3}/r_1^{1/3}$,

$$\mathbb{E}|g_i - \nabla F_\delta(y)_i| \leq 3000\sqrt{\frac{2n\epsilon_1 pL}{r_1}} \leq 3000L\sqrt{\frac{2n\epsilon_1 p}{r_1}} = \frac{3000\sqrt{2}Ln^{2/3}\epsilon_1^{1/3}}{r_1^{1/3}}; \tag{2.28}$$

2. With probability at most $n/p = n^{2/3}\epsilon_1^{1/3}/r_1^{1/3}$, the algorithm fails. From Lipschitz continuity, $|\nabla F_\delta(x)_i| \leq L$, and from Lemma 2.3, $|g_i| \leq L$. Therefore,

$$\mathbb{E}|g_i - \nabla F_\delta(y)_i| \leq 2L. \tag{2.29}$$

Finally, we have

$$\mathbb{E}_{y \sim G_{B_\infty(x,r_1),l}}|g_i - \nabla F_\delta(y)_i| \leq \frac{3000\sqrt{2}Ln^{2/3}\epsilon_1^{1/3}}{r_1^{1/3}} + \frac{2Ln}{p} < \frac{5000Ln^{2/3}\epsilon_1^{1/3}}{r_1^{1/3}}, \tag{2.30}$$

hence

$$\mathbb{E}_{y \sim G_{B_\infty(x,r_1),l}}\|g - \nabla F_\delta(y)\|_1 \leq \frac{5000Ln^{5/3}\epsilon_1^{1/3}}{r_1^{1/3}}. \tag{2.31}$$

Thus from Lemma 2.4,

$$f(q) \geq f(x) + \langle g, q - x\rangle - \zeta\|q - x\|_\infty - 4nr_1L - 2L\delta \tag{2.32}$$

for all $q \in \mathbb{R}^n$ where $\mathbb{E}\zeta \leq \frac{5000Ln^{5/3}\epsilon_1^{1/3}}{r_1^{1/3}}$. Now let $\delta \to 0$. Then $F_\delta \to f$, $\epsilon_1 \to \epsilon$, and $g \to \tilde{g}$. Finally,

$$f(q) \geq f(x) + \langle \tilde{g}, q - x\rangle - \zeta\|q - x\|_\infty - 4nr_1L \tag{2.33}$$

for all $q \in \mathbb{R}^n$, where $\mathbb{E}\zeta \leq \frac{5000Ln^{5/3}\epsilon^{1/3}}{r_1^{1/3}}$. □

2.3 Membership to separation

In this subsection we show how the approximate subgradient procedure (Algorithm 3) fits into the reduction from separation to membership presented in [21]. We use the *height function* $h_p \colon \mathbb{R}^n \to \mathbb{R}$ defined in [21] for any vector $p \in \mathbb{R}^n$, as

$$h_p(x) = -\max\{t \in \mathbb{R} \mid x + t\hat{p} \in K\}, \tag{2.34}$$

where $\hat{p}$ is the unit vector in the direction of p. The height function has the following properties:

Proposition 2.2 (Lemmas 11 and 12 of [21])**.** *Let $K \subset \mathbb{R}^n$ be a convex set with $B_2(0,r) \subseteq K \subseteq B_2(0,R)$ for some $R > r > 0$. Then for any $p \in \mathbb{R}^n$, the height function (2.34) satisfies*

(i) *$h_p(x)$ is convex,*
(ii) *$h_p(x) \leq 0$ for all $x \in K$, and*
(iii) *for all $\delta > 0$, $h_p(x)$ is $\frac{R+\delta}{r-\delta}$-Lipschitz continuous for $x \in B_2(0,\delta)$.*

Algorithm 4: `SeparatingHalfspace`(K, p, ρ, δ)

Data: Convex set K such that $B_2(0,r) \subset K \subset B_2(0,R), \kappa = R/r$, δ-precision membership oracle for K, point p.

1 **if** *the membership oracle asserts that* $p \in B_2(K,\delta)$ **then**

2 **Output:** $p \in B_2(K,\delta)$.

3 **else if** $p \notin B_2(0,R)$ **then**

4 **Output:** the halfspace $\{x \in \mathbb{R}^n \mid 0 > \langle x-p, p\rangle\}$.

5 **else**

6 Define $h_p(x)$ as in (2.34). The evaluation oracle for $h_p(x)$ for any $x \in B(0,r/2)$ can be implemented to precision $\epsilon = 7\kappa\delta$ using $\log(1/\epsilon)$ queries to the membership oracle for K;

7 Compute $\tilde{g} = \texttt{QuantumSubgradient}(h_p, \epsilon, L, 0, n\epsilon^{1/2})$;

8 **Output:** the halfspace $\{x \in \mathbb{R}^n \mid (30000R+25)\, n^3\epsilon^{1/6}\kappa^2/\rho \geq \langle \tilde{g}, x-p\rangle\}$.

Now we are ready to analyze Algorithm 4.

Theorem 2.3. *Let $K \subset \mathbb{R}^n$ be a convex set such that $B_2(0,r) \subseteq K \subseteq B_2(0,R)$ for some $R > r > 0$. Let $\rho \in (0,1), \kappa = R/r$ and $\delta \in (0, \min\{r/7\kappa, 1/7\kappa\})$. Then with probability at least $1-\rho$,* `SeparatingHalfspace`(K,p,ρ,δ) *outputs a halfspace that contains K and not p.*

Proof. Since $\delta \leq \min\{r/7\kappa, 1/7\kappa\}$, $\epsilon \leq \min\{1, r\}$. If $p \in B_2(K,\delta)$, the algorithm is trivially correct. If $p \notin B_2(0,R)$, the algorithm outputs a halfspace that contains $B_2(0,R)$ (and therefore contains K), and not p.

Finally, suppose $p \notin B_2(K,-\delta)$ and $p \in B_2(0,R)$. Since $\epsilon \geq \delta$, $p \notin B_2(K,-\epsilon)$. The height function $h_p(x)$ is 3κ-Lipschitz for all $x \in B_2(0,r/2)$, where $\kappa := R/r$. Define $r_1 = n\epsilon^{1/2}$. Since $\epsilon < \min\{1, r_1/n\}$, Theorem 2.2 implies

$$h_p(x) \geq h_p(0) + \langle \tilde{g}, x\rangle - \zeta\|x\|_\infty - 12nr_1\kappa \tag{2.35}$$

for any $x \in K$, where $\mathbb{E}\zeta \leq \frac{15000\kappa n^{4/3}\epsilon^{1/3}}{r_1^{1/3}}$.

Notice that $-p/\kappa \in K$ and $h_p(-p/\kappa) = h_p(0) - \frac{1}{\kappa}\|p\|_2$. From (2.35),

$$h_p(0) - \frac{1}{\kappa}\|p\|_2 \geq h_p(0) + \langle \tilde{g}, -p/\kappa\rangle - \frac{1}{\kappa}\zeta\|p\|_\infty - 12nr_1\kappa, \tag{2.36}$$

hence

$$\langle \tilde{g}, p\rangle \geq \|p\|_2 - \zeta\|p\|_\infty - 12nr_1\kappa^2. \tag{2.37}$$

As claimed in Line 6 of Algorithm 4, $h_p(x)$ can be evaluated with any precision ϵ such that $7\kappa\delta \leq \epsilon$ using $O(\log(1/\epsilon))$ queries to a membership oracle with error δ; the proof is deferred to Lemma B.3.

Since the membership oracle returns a negative response $p \notin B_2(K,-\delta)$, the error ϵ in $h_p(x)$ must be $\geq \delta$, and hence $p \notin B_2(K,-\epsilon)$. We are also given that $B_2(0,r) \subseteq K$. As a result, we have $(1-\frac{\epsilon}{r})\, K \subseteq B_2(K,-\epsilon)$. Thus,

$$h_p(0) \geq -\left(1-\frac{\epsilon}{r}\right)\|p\|_2 \geq -\|p\|_2 + \epsilon\kappa. \tag{2.38}$$

From (2.35), (2.36), and (2.38), we have

$$h_p(x) \geq \langle \tilde{g}, x-p\rangle - \zeta\|x\|_\infty - \zeta\|p\|_\infty - 12nr_1\kappa - 12nr_1\kappa^2 - \epsilon\kappa \tag{2.39}$$
$$\geq \langle \tilde{g}, x-p\rangle - 2\zeta R - 24nr_1\kappa^2 - \epsilon\kappa, \tag{2.40}$$

so $\langle \tilde{g}, x-p\rangle \leq \tilde{\zeta}$ for all $x \in K$, where

$$\mathbb{E}\tilde{\zeta} \leq \frac{30000Rn^{5/3}\epsilon^{1/3}\kappa}{r_1^{1/3}} + 24nr_1\kappa^2 + \epsilon\kappa \tag{2.41}$$
$$\leq 30000Rn\epsilon^{1/6}\kappa + 24n^3\epsilon^{1/2}\kappa^2 + \epsilon\kappa \tag{2.42}$$
$$\leq (30000R+25)n^3\epsilon^{1/6}\kappa^2. \tag{2.43}$$

Thus the result follows from Markov's inequality. □

Theorem 2.4. *Let $K \subset \mathbb{R}^n$ be a convex set with $B_2(0,r) \subseteq K \subseteq B_2(0,R)$ and $\kappa = R/r$ for some $R > r > 0$, and let $\eta > 0$ be fixed. Further suppose that $R, r, \kappa = \mathrm{poly}(n)$. Then a separating oracle for K with error η can be implemented using $\tilde{O}(1)$ queries to a membership oracle for K and $\tilde{O}(n)$ gates.*

Proof. Clearly, the unit vector in the direction $\tilde{g}$ (from Algorithm 4) determines a separating hyperplane given a point $p \notin B_2(K, -\epsilon)$. From (2.37), we have

$$\langle \tilde{g}, p\rangle \geq \|p\|_2 - \left(\frac{15000n^{5/3}\epsilon^{1/3}\kappa}{r_1}\right)\|p\|_\infty. \tag{2.44}$$

Letting $\frac{15000n^{5/3}\epsilon^{1/3}}{r_1} < \frac{1}{2\kappa^2}$, we have

$$\|\tilde{g}\|_2 R \geq r - \frac{R}{2\kappa} \;\Rightarrow\; \|\tilde{g}\|_2 \geq \frac{1}{2\kappa}. \tag{2.45}$$

Thus, we have a separating oracle with error margin $(60000R+50)\,n^3\epsilon^{1/6}\kappa^3\rho^{-1}$ and failure probability ρ. Setting $\rho = \left((60000R+50)\,n^3\epsilon^{1/6}\kappa^3\right)^{1/2}$, we have a composite error of $\left((60000R+50)\,n^3\epsilon^{1/6}\kappa^3\right)^{1/2}$. To have error at most η, we take $\epsilon \leq \eta^6/((60000R+15)^6 n^{18}\kappa^{18})$.

We finally obtain

$$\delta = \frac{\epsilon}{7\kappa} \leq \frac{1}{7\kappa}\min\left\{\frac{\eta^6}{(60000R+50)^6\,n^{18}\kappa^{18}}, \frac{1}{8\kappa^6\left(\frac{15000n^{5/3}\epsilon^{1/3}}{r_1}\right)^3}, r, 1\right\}. \tag{2.46}$$

Consequently, we have $\mathrm{SEP}_\eta = \tilde{O}(1)\,\mathrm{MEM}_\delta$, where

$$\delta = \frac{\epsilon}{7\kappa} \leq \frac{1}{7\kappa}\min\left\{\frac{\eta^6}{(60000R+50)^6\,n^{18}\kappa^{18}}, \frac{1}{8\kappa^6\left(\frac{15000n^{5/3}\epsilon^{1/3}}{r_1}\right)^3}, r, 1\right\}. \tag{2.47}$$

Therefore, $1/\epsilon$ and $1/\delta$ are both $O(\mathrm{poly}(n))$. Implementing the evaluation oracle takes $\mathrm{poly}(\log(1/\epsilon))$ membership queries and a further $\tilde{O}(1)$ queries are used for the subgradient.

The evaluation requires $\tilde{O}(1/\epsilon)$ gates and the `SmoothQuantumGradient` uses $n\,\mathrm{poly}(\log(1/\epsilon))$ gates. Thus a total of $\mathrm{poly}(\log(1/\eta))$ queries and $n\,\mathrm{poly}(\log(1/\eta))$ gates are used. □

2.4 Separation to optimization

It is known that an optimization oracle for a convex set can be implemented in $\tilde{O}(n)$ queries to a separation oracle. Specifically, Theorem 15 of [21] states:

Theorem 2.5 (Separation to Optimization)**.** *Let K be a convex set satisfying $B_2(0,r) \subset K \subset B_2(0,R)$ and let $\kappa = 1/r$. For any $0 < \epsilon < 1$, with probability $1-\epsilon$, we can compute $x \in B_2(K,\epsilon)$ such that $c^T x \leq \min_{x \in K} c^T x + \epsilon \|c\|_2$, using $O(n \log(n\kappa/\epsilon))$ queries to $\mathrm{SEP}_\eta(K)$, where $\eta = \mathrm{poly}(\epsilon/n\kappa)$, and $\tilde{O}(n^3)$ arithmetic operations.*

From Theorem 2.5 and Theorem 2.4, we have the following result

Theorem 2.6 (Membership to Optimization)**.** *Let K be a convex set satisfying $B_2(0,r) \subset K \subset B_2(0,R)$ and let $\kappa = 1/r$. For any $0 < \epsilon < 1$, with probability $1-\epsilon$, we can compute $x \in B_2(K,\epsilon)$ such that $c^T x \leq \min_{x \in K} c^T x + \epsilon$, using $\tilde{O}(n)$ queries to a membership oracle for K with error δ, where $\delta = O(\mathrm{poly}(\epsilon))$, and $\tilde{O}(n^3)$ gates.*

Proof. Using Theorem 2.4 with $\eta = \mathrm{poly}(\epsilon/n\kappa)$, each query to the separation oracle requires $\tilde{O}(1)$ queries to a membership oracle with error $\delta = O(\mathrm{poly}(\epsilon))$. We make $\tilde{O}(n)$ separation queries and perform a further $\tilde{O}(n^3)$ arithmetic operations, so the result follows. □

Theorem 2.1 follows directly from Theorem 2.6.

3 Lower bound

In this section, we prove our quantum lower bound on convex optimization (Theorem 1.2). We prove separate lower bounds on membership queries (Section 3.1) and evaluation queries (Section 3.2). We then combine these lower bounds into a single optimization problem in Section 3.3, establishing Theorem 1.2.

3.1 Membership queries

In this subsection, we establish a membership query lower bound using a reduction from the following search-with-wildcards problem:

Theorem 3.1 ([1, Theorem 1])**.** *For any $s \in \{0,1\}^n$, let O_s be a wildcard oracle satisfying*

$$O_s|T\rangle|y\rangle|0\rangle = |T\rangle|y\rangle|Q_s(T,y)\rangle \tag{3.1}$$

for all $T \subseteq [n]$ and $y \in \{0,1\}^{|T|}$, where $Q_s(T,y) = \delta[s_{|T} = y]$. Then the bounded-error quantum query complexity of determining s is $O(\sqrt{n}\log n)$ and $\Omega(\sqrt{n})$.

We use Theorem 3.1 to give an $\Omega(\sqrt{n})$ lower bound on membership queries for convex optimization. Specifically, we consider the following *sum-of-coordinates optimization problem*:

Definition 3.1. *Let*

$$\mathcal{C}_s := \bigtimes_{i=1}^{n} [s_i - 2, s_i + 1], \qquad s_i \in \{0,1\} \quad \forall\, i \in [n], \tag{3.2}$$

where $\times$ *is the Cartesian product on different coordinates. In the* sum-of-coordinates optimization *problem, the goal is to minimize*

$$f(x) = \sum_{i\in[n]} x_i \quad s.t.\ x \in \mathcal{C}_s. \tag{3.3}$$

Intuitively, Definition 3.1 concerns an optimization problem on a hypercube where the function is simply the sum of the coordinates, but the position of the hypercube is unknown. Note that the function f in (3.3) is convex and 1-Lipschitz continuous.

We prove the hardness of solving sum-of-coordinates optimization using its membership oracle:

Theorem 3.2. *Given an instance of the sum-of-coordinates optimization problem with membership oracle* $O_{\mathcal{C}_s}$*, it takes* $\Omega(\sqrt{n})$ *quantum queries to* $O_{\mathcal{C}_s}$ *to output an* $\tilde{x} \in \mathcal{C}_s$ *such that*

$$f(\tilde{x}) \leq \min_{x\in\mathcal{C}_s} f(x) + \frac{1}{3}, \tag{3.4}$$

with success probability at least 0.9*.*

Proof. Assume that we are given an arbitrary string $s \in \{0,1\}^n$ together with the membership oracle $O_{\mathcal{C}_s}$ for the sum-of-coordinates optimization problem.

We prove that a quantum query to $O_{\mathcal{C}_s}$ can be simulated by a quantum query to the oracle O_s in (3.1) for search with wildcards. Consider an arbitrary point $x \in \mathbb{R}^n$ in the sum-of-coordinates problem. We partition $[n]$ into four sets:

$$T_{x,0} := \{i \in [n] \mid x_i \in [-2,-1)\} \tag{3.5}$$
$$T_{x,1} := \{i \in [n] \mid x_i \in (1,2]\} \tag{3.6}$$
$$T_{x,\text{mid}} := \{i \in [n] \mid x_i \in [-1,1]\} \tag{3.7}$$
$$T_{x,\text{out}} := \{i \in [n] \mid |x_i| > 2\}, \tag{3.8}$$

and denote $T_x := T_{x,0} \cup T_{x,1}$ and $y^{(x)} \in \{0,1\}^{|T_x|}$ such that

$$y_i^{(x)} = \begin{cases} 0 & \text{if } i \in T_{x,0} \\ 1 & \text{if } i \in T_{x,1}. \end{cases} \tag{3.9}$$

We prove that $O_{\mathcal{C}_s}(x) = Q_s(T_x, y^{(x)})$ if $T_{x,\text{out}} = \varnothing$, and $O_{\mathcal{C}_s}(x) = 0$ otherwise. On the one hand, if $O_{\mathcal{C}_s}(x) = 1$, we have $x \in \mathcal{C}_s$. Because for all $i \in [n]$, $x_i \in [s_i - 2, s_i + 1] \subset [-2, 2]$ for both $s_i = 0$ and $s_i = 1$, we must have $T_{x,\text{out}} = \varnothing$. Now consider any $i \in T_x$. If $i \in T_{x,0}$, then $x_i \in [-2,-1)$. Because $x_i \in [0-2, 0+1]$ and $x_i \notin [1-2, 1+1]$, we must have $s_i = 0$ since $x_i \in [s_i - 2, s_i + 1]$. Similarly, if $i \in T_{x,1}$, then we must have $s_i = 1$. As a result of (3.9), for all $i \in T_x$ we have $s_i = y_i^{(x)}$; in other words, $s_{|T_x} = y^{(x)}$ and $Q_s(T_x, y^{(x)}) = 1 = O_{\mathcal{C}_s}(x)$.

On the other hand, if $O_{\mathcal{C}_s}(x) = 0$, there exists an $i_0 \in [n]$ such that $x_{i_0} \notin [s_{i_0} - 2, s_{i_0} + 1]$. We must have $i_0 \notin T_{x,\text{mid}}$ because $[-1,1] \subset [s_{i_0} - 2, s_{i_0} + 1]$ regardless of whether $s_{i_0} = 0$ or $s_{i_0} = 1$. Next, if $i_0 \in T_{x,\text{out}}$, then $T_{x,\text{out}} \neq \varnothing$ and we correctly obtain $O_{\mathcal{C}_s}(x) = 0$. The remaining cases are $i_0 \in T_{x,0}$ and $i_0 \in T_{x,1}$. If $i_0 \in T_{x,0}$, because $x_{i_0} \in [-2,-1) \subset [0-2, 0+1]$ and $x_{i_0} \notin [s_{i_0} - 2, s_{i_0} + 1]$, we must have $s_{i_0} = 1$, and thus $s_{|T_x} \neq y^{(x)}$ because $y_{i_0}^{(x)} = 0$ by

(3.9). If $i_0 \in T_{x,1}$, we similarly have $s_{i_0} = 0$, $y_{i_0}^{(x)} = 1$, and thus $s_{|T_x} \neq y^{(x)}$. In both cases, $s_{|T_x} \neq y^{(x)}$, so $Q_s(T_x, y^{(x)}) = 0 = O_{\mathcal{C}_s}(x)$.

Therefore, we have established that $O_{\mathcal{C}_s}(x) = Q_s(T_x, y^{(x)})$ if $T_{x,\text{out}} = \varnothing$, and $O_{\mathcal{C}_s}(x) = 0$ otherwise. In other words, a quantum query to $O_{\mathcal{C}_s}$ can be simulated by a quantum query to O_s.

We next prove that a solution $\tilde{x}$ of the sum-of-coordinates problem satisfying (3.4) solves the search-with-wildcards problem in Theorem 3.1. Because $\min_{x \in \mathcal{C}_s} f(x) = \sum_{i=1}^n (s_i - 2)$, we have

$$f(\tilde{x}) = \sum_{i=1}^n \tilde{x}_i \leq \frac{1}{3} + \sum_{i=1}^n (s_i - 2). \tag{3.10}$$

On the one hand, for all $j \in [n]$ we have $\tilde{x}_j \geq s_j - 2$ since $\tilde{x} \in \mathcal{C}_s$; on the other hand, by (3.10) we have

$$\frac{1}{3} + \sum_{i=1}^n (s_i - 2) \geq \sum_{i=1}^n \tilde{x}_i \geq \tilde{x}_j + \sum_{i \in [n],\ i \neq j} (s_i - 2), \tag{3.11}$$

which implies $\tilde{x}_j \leq s_j - 2 + \frac{1}{3}$. In all,

$$\tilde{x}_i \in [s_i - 2, s_i - 2 + \tfrac{1}{3}] \quad \forall\, i \in [n]. \tag{3.12}$$

Define a rounding function $\operatorname{sgn}_{-3/2} \colon \mathbb{R} \to \{0, 1\}$ as

$$\operatorname{sgn}_{-3/2}(z) = \begin{cases} 0 & \text{if } z < -3/2 \\ 1 & \text{otherwise.} \end{cases} \tag{3.13}$$

We prove that $\operatorname{sgn}_{-3/2}(\tilde{x}) = s$ (here $\operatorname{sgn}_{-3/2}$ is applied on all n coordinates, respectively). For all $i \in [n]$, if $s_i = 0$, then $\tilde{x}_i \in [-2, -\frac{5}{3}] \subset (-\infty, -\frac{3}{2})$ by (3.12), which implies $\operatorname{sgn}_{-3/2}(\tilde{x}_i) = 0$ by (3.13). Similarly, if $s_i = 1$, then $\tilde{x}_i \in [-1, -\frac{2}{3}] \subset (-\frac{3}{2}, +\infty)$ by (3.12), which implies $\operatorname{sgn}_{-3/2}(\tilde{x}_i) = 1$ by (3.13).

In all, if we can solve the sum-of-coordinates optimization problem with an $\tilde{x}$ satisfying (3.4), we can solve the search-with-wildcards problem. By Theorem 3.2, the search-with-wildcards problem has quantum query complexity $\Omega(\sqrt{n})$; since a query to the membership oracle $O_{\mathcal{C}_s}$ can be simulated by a query to the wildcard oracle O_s, we have established an $\Omega(\sqrt{n})$ quantum lower bound on membership queries to solve the sum-of-coordinates optimization problem. □

3.2 Evaluation queries

In this subsection, we establish an evaluation query lower bound by considering the following *max-norm optimization problem*:

Definition 3.2. *In the* max-norm optimization problem*, the goal is to minimize a function* $f_c \colon \mathbb{R}^n \to \mathbb{R}$ *satisfying*

$$f_c(x) = \max_{i \in [n]} |\pi(x_i) - c_i| + \Big(\sum_{i=1}^n |\pi(x_i) - x_i| \Big) \tag{3.14}$$

for some $c \in \{0,1\}^n$*, where* $\pi\colon \mathbb{R} \to [0,1]$ *is defined as*

$$\pi(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \le x \le 1 \\ 1 & \text{if } x > 1. \end{cases} \tag{3.15}$$

Observe that for all $x \in [0,1]^n$, we have $f_c(x) = \max_{i\in[n]} |x_i - c_i|$. Intuitively, Definition 3.2 concerns an optimization problem under the max-norm (i.e., L_∞ norm) distance from c for all x in the unit hypercube $[0,1]^n$; for all x not in the unit hypercube, the optimizing function pays a penalty of the L_1 distance between x and its projection $\pi(x)$ onto the unit hypercube. The function f_c is 2-Lipschitz continuous with a unique minimum at $x = c$; we prove in Lemma C.1 that f_c is convex.

We prove the hardness of solving max-norm optimization using its evaluation oracle:

Theorem 3.3. *Given an instance of the max-norm optimization problem with an evaluation oracle* O_{f_c}*, it takes* $\Omega(\sqrt{n}/\log n)$ *quantum queries to* O_{f_c} *to output an* $\tilde{x} \in [0,1]^n$ *such that*

$$f_c(\tilde{x}) \le \min_{x\in[0,1]^n} f_c(x) + \frac{1}{3}, \tag{3.16}$$

with success probability at least 0.9*.*

The proof of Theorem 3.3 has two steps. First, we prove a weaker lower bound with respect to the precision of the evaluation oracle:

Lemma 3.1. *Suppose we are given an instance of the max-norm optimization problem with an evaluation oracle* O_{f_c} *that has precision* $0 < \delta < 0.05$*, i.e.,* f_c *is provided with* $\lceil \log_2(1/\delta) \rceil$ *bits of precision. Then it takes* $\Omega(\sqrt{n}/\log(1/\delta))$ *quantum queries to* O_{f_c} *to output an* $\tilde{x} \in [0,1]^n$ *such that*

$$f_c(\tilde{x}) \le \min_{x\in[0,1]^n} f_c(x) + \frac{1}{3}, \tag{3.17}$$

with success probability at least 0.9*.*

The second step simulates a perfectly precise query to f_c by a rough query:

Lemma 3.2. *One classical (resp., quantum) query to* O_{f_c} *with perfect precision can be simulated by one classical query (resp., two quantum queries) to* O_{f_c} *with precision* $1/5n$*.*

Theorem 3.3 simply follows from the two propositions above: by Lemma 3.2, we can assume that the evaluation oracle O_{f_c} has precision $1/5n$, so Lemma 3.1 implies that it takes $\Omega(\sqrt{n}/\log 5n) = \Omega(\sqrt{n}/\log n)$ quantum queries to O_{f_c} to output an $\tilde{x} \in [0,1]^n$ satisfying (3.16) with success probability 0.9.

The proofs of Lemma 3.1 and Lemma 3.2 are given in Section 3.2.1 and Section 3.2.2, respectively.

3.2.1 $\tilde{\Omega}(\sqrt{n})$ quantum lower bound on a low-precision evaluation oracle

Similar to the proof of Theorem 3.2, we also use Theorem 3.1 (the quantum lower bound on search with wildcards) to give a quantum lower bound on the number of evaluation queries required to solve the max-norm optimization problem.

Proof of Lemma 3.1. Assume that we are given an arbitrary string $c \in \{0,1\}^n$ together with the evaluation oracle O_{f_c} for the max-norm optimization problem. To show the lower bound, we reduce the search-with-wildcards problem to the max-norm optimization problem.

We first establish that an evaluation query to O_f can be simulated using wildcard queries on c. Notice that if we query an arbitrary $x \in \mathbb{R}^n$, by (3.14) we have

$$f_c(x) = \max_{i\in[n]} |\pi(x_i) - c_i| + \Big(\sum_{i=1}^{n} |\pi(x_i) - x_i|\Big) = f_c(\pi(x)) + \Big(\sum_{i=1}^{n} |\pi(x_i) - x_i|\Big) \tag{3.18}$$

where $\pi(x) := (\pi(x_1), \ldots, \pi(x_n))$. In particular, the difference of $f_c(x)$ and $f_c(\pi(x))$ is an explicit function of x that is independent of c. Thus the query $O_{f_c}(x)$ can be simulated using one query to $O_{f_c}(\pi(x))$ where $\pi(x) \in [0,1]^n$. It follows that we can restrict ourselves without loss of generality to implementing evaluation queries for $x \in [0,1]^n$.

Now we consider a decision version of oracle queries to f_c, denoted $O_{f_{c,\mathrm{dec}}}$, where the function $f_{c,\mathrm{dec}}\colon [0,1]^n \times [0,1] \to \{0,1\}$ satisfies

$$f_{c,\mathrm{dec}}(x,t) = \delta[f_c(x) \le t]. \tag{3.19}$$

(We restrict to $t \in [0,1]$ because $f_c(x) \in [0,1]$ always holds for $x \in [0,1]^n$.) Using binary search, a query to O_{f_c} with precision δ can be simulated by at most $\lceil \log_2(1/\delta) \rceil = O(\log 1/\delta)$ queries to the oracle $O_{f_{c,\mathrm{dec}}}$.

Next, we prove that a query to $O_{f_{c,\mathrm{dec}}}$ can be simulated by a query to the search-with-wildcards oracle O_c in (3.1). Consider an arbitrary query $(x,t) \in [0,1]^n \times [0,1]$ to $O_{f_{c,\mathrm{dec}}}$. For convenience, we denote $J_{0,t} := [0,t]$, $J_{1,t} := [1-t,1]$, and

$$I_{0,t} := J_{0,t} - (J_{0,t} \cap J_{1,t}) \tag{3.20}$$
$$I_{1,t} := J_{1,t} - (J_{0,t} \cap J_{1,t}) \tag{3.21}$$
$$I_{\mathrm{mid},t} := J_{0,t} \cap J_{1,t} \tag{3.22}$$
$$I_{\mathrm{out},t} := [0,1] - (J_{0,t} \cup J_{1,t}). \tag{3.23}$$

We partition $[n]$ into four sets:

$$T_{x,0,t} := \{i \in [n] \mid x_i \in I_{0,t}\} \tag{3.24}$$
$$T_{x,1,t} := \{i \in [n] \mid x_i \in I_{1,t}\} \tag{3.25}$$
$$T_{x,\mathrm{mid},t} := \{i \in [n] \mid x_i \in I_{\mathrm{mid},t}\} \tag{3.26}$$
$$T_{x,\mathrm{out},t} := \{i \in [n] \mid x_i \in I_{\mathrm{out},t}\}. \tag{3.27}$$

The strategy here is similar to the proof of Theorem 3.2: $T_{x,\mathrm{mid},t}$ corresponds to the coordinates such that $|x_i - c_i| \le t$ regardless of whether $c_i = 0$ or 1 (and hence c_i does not influence whether or not $\max_{i\in[n]} |x_i - c_i| \le t$); $T_{x,\mathrm{out},t}$ corresponds to the coordinates such that $|x_i - c_i| > t$ regardless of whether $c_i = 0$ or 1 (so $\max_{i\in[n]} |x_i - c_i| > t$ provided $T_{x,\mathrm{out},t}$

is nonempty); and $T_{x,0,t}$ (resp., $T_{x,1,t}$) corresponds to the coordinates such that $|x_i - c_i| \le t$ only when $c_i = 0$ (resp., $c_i = 1$).

Denote $T_{x,t} := T_{x,0,t} \cup T_{x,1,t}$ and let $y^{(x,t)} \in \{0,1\}^{|T_{x,t}|}$ such that

$$y_i^{(x,t)} = \begin{cases} 0 & \text{if } i \in T_{x,0,t} \\ 1 & \text{if } i \in T_{x,1,t}. \end{cases} \tag{3.28}$$

We will prove that $O_{f_{c,\mathrm{dec}}}(x) = Q_c(T_{x,t}, y^{(x,t)})$ if $T_{x,\mathrm{out},t} = \varnothing$, and $O_{f_{c,\mathrm{dec}}}(x) = 0$ otherwise.

On the one hand, if $O_{f_{c,\mathrm{dec}}}(x) = 1$, we have $f_c(x) \le t$. In other words, for all $i \in [n]$ we have $|x_i - c_i| \le t$, which implies

$$x_i \in J_{c_i,t} \quad \forall\, i \in [n]. \tag{3.29}$$

Since $J_{c_i,t} \subseteq J_{0,t} \cup J_{1,t}$, we have $x_i \in J_{0,t} \cup J_{1,t}$ for all $i \in [n]$, and thus $T_{x,\mathrm{out},t} = \varnothing$ by (3.23) and (3.27). Now consider any $i \in T_{x,t}$. If $i \in T_{x,0,t}$, then $x_i \in I_{0,t}$ by (3.24). By (3.20) we have $x_i \in J_{0,t}$ and $x_i \notin J_{1,t}$, and thus $c_i = 0$ by (3.29). Similarly, if $i \in T_{x,1,t}$, then we must have $c_i = 1$. As a result of (3.28), for all $i \in T_{x,t}$ we have $c_i = y_i^{(x,t)}$; in other words, $c_{|T_{x,t}} = y^{(x,t)}$ and $Q_c(T_{x,t}, y^{(x,t)}) = 1 = O_{f_{c,\mathrm{dec}}}(x)$.

On the other hand, if $O_{f_{c,\mathrm{dec}}}(x) = 0$, there exists an $i_0 \in [n]$ such that

$$x_{i_0} \notin J_{c_{i_0},t}. \tag{3.30}$$

Therefore, we must have $i_0 \notin T_{x,\mathrm{mid},t}$ since (3.22) implies $I_{\mathrm{mid},t} = J_{0,t} \cap J_{1,t} \subseteq J_{c_{i_0},t}$. Next, if $i_0 \in T_{x,\mathrm{out},t}$, then $T_{x,\mathrm{out},t} \neq \varnothing$ and we correctly obtain $O_{f_{c,\mathrm{dec}}}(x) = 0$. The remaining cases are $i_0 \in T_{x,0,t}$ and $i_0 \in T_{x,1,t}$.

If $i_0 \in T_{x,0,t}$, then $y_{i_0}^{(x,t)} = 0$ by (3.28). By (3.24) we have $x_{i_0} \in I_{0,t}$, and by (3.20) we have $x_{i_0,t} \in J_{0,t}$ and $x_{i_0} \notin J_{1,t}$; therefore, we must have $c_{i_0} = 1$ by (3.30). As a result, $c_{|T_{x,t}} \neq y^{(x,t)}$ at i_0. If $i_0 \in T_{x,1,t}$, we similarly have $c_{i_0} = 0$, $y_{i_0}^{(x,t)} = 1$, and thus $c_{|T_{x,t}} \neq y^{(x,t)}$ at i_0. In either case, $c_{|T_{x,t}} \neq y^{(x,t)}$, and $Q_c(T_{x,t}, y^{(x,t)}) = 0 = O_{f_{c,\mathrm{dec}}}(x)$.

Therefore, we have established that $O_{f_{c,\mathrm{dec}}}(x) = Q_c(T_{x,t}, y^{(x,t)})$ if $T_{x,\mathrm{out},t} = \varnothing$, and $O_{f_{c,\mathrm{dec}}}(x) = 0$ otherwise. In other words, a quantum query to $O_{f_{c,\mathrm{dec}}}$ can be simulated by a quantum query to the search-with-wildcards oracle O_c. Together with the fact that a query to O_{f_c} with precision δ can be simulated by $O(\log 1/\delta)$ queries to $O_{f_{c,\mathrm{dec}}}$, it can also be simulated by $O(\log 1/\delta)$ queries to O_c.

We next prove that a solution $\tilde{x}$ of the max-norm optimization problem satisfying (3.17) solves the search-with-wildcards problem in Theorem 3.1. Because $\min_{x \in [0,1]^n} f_c(x) = 0$, considering the precision of at most $\delta < 0.05$ we have

$$f_c(\tilde{x}) \le \tfrac{1}{3} + \delta \le 0.4. \tag{3.31}$$

In other words,

$$\tilde{x}_i \in [c_i - 0.4, c_i + 0.4] \quad \forall\, i \in [n]. \tag{3.32}$$

Similar to (3.13), we define a rounding function $\mathrm{sgn}_{1/2} \colon \mathbb{R} \to \{0,1\}$ as

$$\mathrm{sgn}_{1/2}(z) = \begin{cases} 0 & \text{if } z < 1/2 \\ 1 & \text{otherwise.} \end{cases} \tag{3.33}$$

We prove that $\mathrm{sgn}_{1/2}(\tilde{x}) = c$ (here $\mathrm{sgn}_{1/2}$ is applied coordinate-wise). For all $i \in [n]$, if $c_i = 0$, then $\tilde{x}_i \in [0, 0.4] \subset (-\infty, 1/2)$ by (3.32), which implies $\mathrm{sgn}_{1/2}(\tilde{x}_i) = 0$ by (3.33). Similarly, if $c_i = 1$, then $\tilde{x}_i \in [0.6, 1] \subset (1/2, +\infty)$ by (3.32), which implies $\mathrm{sgn}_{1/2}(\tilde{x}_i) = 1$ by (3.33).

We have shown that if we can solve the max-norm optimization problem with an $\tilde{x}$ satisfying (3.17), we can solve the search-with-wildcards problem. By Theorem 3.2, the search-with-wildcards problem has quantum query complexity $\Omega(\sqrt{n})$; since a query to the evaluation oracle O_{f_c} can be simulated by $O(\log 1/\delta)$ queries to the wildcard oracle O_c, we have established an $\Omega(\sqrt{n}/\log(1/\delta))$ quantum lower bound on the number of evaluation queries needed to solve the max-norm optimization problem. □

3.2.2 Discretization: simulating perfectly precise queries by low-precision queries

In this subsection we prove Lemma 3.2, which we rephrase more formally as follows. Throughout this subsection, the function f_c in (3.14) is abbreviated by f for notational convenience.

Lemma 3.3. *Assume that $\hat{f}\colon [0,1]^n \to [0,1]$ satisfies $|\hat{f}(x) - f(x)| \le \frac{1}{5n}\ \forall\, x \in [0,1]^n$. Then one classical (resp., quantum) query to O_f can be simulated by one classical query (resp., two quantum queries) to $O_{\hat{f}}$.*

To achieve this, we present an approach that we call *discretization*. Instead of considering queries on all of $[0,1]^n$, we only consider a discrete subset $D_n \subseteq [0,1]^n$ defined as

$$D_n := \{\chi(a,\pi) \mid a \in \{0,1\}^n \text{ and } \pi \in S_n\}, \tag{3.34}$$

where S_n is the symmetric group on $[n]$ and $\chi\colon \{0,1\}^n \times S_n \to [0,1]^n$ satisfies

$$\chi(a,\pi)_i = (1-a_i)\tfrac{\pi(i)}{2n+1} + a_i(1 - \tfrac{\pi(i)}{2n+1}) \quad \forall\, i \in [n]. \tag{3.35}$$

Observe that D_n is a subset of $[0,1]^n$.

Since $|S_n| = n!$ and there are 2^n choices for $a \in \{0,1\}^n$, we have $|D_n| = 2^n n!$. For example, when $n = 2$, we have

$$D_2 = \{(\tfrac{1}{5},\tfrac{2}{5}), (\tfrac{1}{5},\tfrac{3}{5}), (\tfrac{4}{5},\tfrac{2}{5}), (\tfrac{4}{5},\tfrac{3}{5}), (\tfrac{2}{5},\tfrac{1}{5}), (\tfrac{2}{5},\tfrac{4}{5}), (\tfrac{3}{5},\tfrac{1}{5}), (\tfrac{3}{5},\tfrac{4}{5})\} \tag{3.36}$$

with $|D_2| = 2^2 \cdot 2! = 8$.

We denote the restriction of the oracle O_f to D_n by $O_{f|D_n}$, i.e.,

$$O_{f|D_n}|x\rangle|0\rangle = |x\rangle|f(x)\rangle \qquad \forall\, x \in D_n. \tag{3.37}$$

In fact, this restricted oracle entirely captures the behavior of the unrestricted function.

Lemma 3.4 (Discretization)**.** *A classical (resp., quantum) query to O_f can be simulated using one classical query (resp., two quantum queries) to $O_{f|D_n}$.*

We prove this proposition by giving an algorithm (Algorithm 5) that performs the simulation. The main idea is to compute $f(x)$ only using x and $f(x^*)$ for some $x^* \in D_n$. We observe that max-norm optimization has the following property: if two strings $x \in [0,1]^n$ and $x^* \in D_n$ are such that $x_1, \dots, x_n, 1-x_1, \dots, 1-x_n$ and $x^*_1, \dots, x^*_n, 1-x^*_1, \dots, 1-x^*_n$ have the same ordering, then

$$\arg\max_{i\in[n]} |x_i - c_i| = \arg\max_{i\in[n]} |x^*_i - c_i|. \tag{3.40}$$

Algorithm 5: Simulate one query to O_f using one query to $O_{f|D_n}$.

Input: $x \in [0,1]^n$;
Output: $f(x) \in [0,1]$;
1 Compute $b \in \{0,1\}^n$ and $\sigma \in S_n$ such that the $2n$ numbers $x_1, x_2, \ldots, x_n, 1-x_1, \ldots, 1-x_n$ are arranged in decreasing order as

$$\begin{aligned} &b_{\sigma(1)}x_{\sigma(1)} + (1-b_{\sigma(1)})(1-x_{\sigma(1)}) \geq \cdots \geq b_{\sigma(n)}x_{\sigma(n)} + (1-b_{\sigma(n)})(1-x_{\sigma(n)}) \\ &\geq (1-b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1-x_{\sigma(n)}) \geq \cdots \geq (1-b_{\sigma(1)})x_{\sigma(1)} + b_{\sigma(1)}(1-x_{\sigma(1)}); \end{aligned} \tag{3.38}$$

2 Compute $x^* \in D_n$ such that $\chi(b, \sigma^{-1}) = x^*$ (where χ is defined in (3.35));
3 Query $f(x^*)$ and let $k^* = (2n+1)(1-f(x^*))$;
4 Return

$$f(x) = \begin{cases} (1-b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1-x_{\sigma(n)}) & \text{if } k^* = n+1 \\ b_{\sigma(k^*)}x_{\sigma(k^*)} + (1-b_{\sigma(k^*)})(1-x_{\sigma(k^*)}) & \text{otherwise.} \end{cases} \tag{3.39}$$

Furthermore, $x^* \in D_n$ ensures that $\{x_1^*, \ldots, x_n^*, 1-x_1^*, \ldots, 1-x_n^*\} = \{\frac{1}{2n+1}, \ldots, \frac{2n}{2n+1}\}$ are $2n$ distinct numbers, so knowing the value of $f(x^*)$ is sufficient to determine the value of the arg max above (denoted i^*) and the corresponding c_{i^*}. We can then recover $f(x) = |x_{i^*} - c_{i^*}|$ using the given value of x. Moreover, $f(x^*)$ is an integer multiple of $\frac{1}{2n+1}$; even if $f(x^*)$ can only be computed with precision $\frac{1}{5n}$, we can round it to the closest integer multiple of $\frac{1}{2n+1}$ which is exactly $f(x^*)$, since the distance $\frac{2n+1}{5n} < \frac{1}{2}$. As a result, we can precisely compute $f(x^*)$ for all $x \in D_n$, and thus we can precisely compute $f(x)$.

We illustrate Algorithm 5 by a simple example. For convenience, we define an order function $\mathrm{Ord}\colon [0,1]^n \to \{0,1\}^n \times S_n$ by $\mathrm{Ord}(x) = (b, \sigma)$ for all $x \in [0,1]^n$, where b and σ satisfy Eq. (3.38).

An example with $n = 3$. Consider the case where the ordering in (3.38) is

$$1 - x_3 \geq x_1 \geq x_2 \geq 1 - x_2 \geq 1 - x_1 \geq x_3. \tag{3.41}$$

Then Algorithm 5 proceeds as follows:

- Line 1: With the ordering (3.41), we have $\sigma(1) = 3$, $\sigma(2) = 1$, $\sigma(3) = 2$; $b_3 = 0$, $b_1 = 1$, $b_2 = 1$.
- Line 2: The point $x^* \in D_3$ that we query given $\mathrm{Ord}(x)$ satisfies $1 - x_3^* = 6/7$, $x_1^* = 5/7$, $x_2^* = 4/7$, $1 - x_2^* = 3/7$, $1 - x_1^* = 2/7$, and $x_3^* = 1/7$; in other words, $x^* = (5/7, 4/7, 1/7)$.
- Line 3: Now we query $f(x^*)$. Since $f(x^*)$ is a multiple of $1/7$ and $f(x^*) \in [1/7, 6/7]$, there are only 6 possibilities: $f(x^*) = 6/7$, $f(x^*) = 5/7$, $f(x^*) = 4/7$, $f(x^*) = 3/7$, $f(x^*) = 2/7$, or $f(x^*) = 1/7$.

  After running Line 1, Line 2, and Line 3, we have a point x^* from the discrete set D_3 such that $\mathrm{Ord}(x) = \mathrm{Ord}(x^*)$. Since they have the same ordering and $|x_i - c_i|$ is either

x_i or $1 - x_i$ for all $i \in [3]$, the function value $f(x^*)$ should essentially reflect the value of $f(x)$; this is made precise in Line 4.

- Line 4: Depending on the value of $f(x^*)$, we have six cases:
  - $f(x^*) = 6/7$: In this case, we must have $c_3 = 1$, so that $|x_3 - c_3| = |1/7 - 1| = 6/7$ ($|x_1 - c_1|$ can only give $5/7$ or $2/7$, and $|x_2 - c_2|$ can only give $4/7$ or $3/7$). Because $1 - x_3$ is the largest in (3.41), we must have $f(x) = 1 - x_3$.
  - $f(x^*) = 5/7$: In this case, we must have $c_1 = 0$, so that $|x_1 - c_1| = |5/7 - 0| = 5/7$. Furthermore, we must have $c_3 = 1$ (otherwise if $c_3 = 0$, $f(x) \geq |x_3 - c_3| = 6/7$). As a result of (3.41), we must have $f(x) = x_1$ since $x_1 \geq x_3$ and $x_1 \geq \max\{x_2, 1 - x_2\}$.
  - $f(x^*) = 4/7$: In this case, we must have $c_2 = 0$, so that $|x_2 - c_2| = |4/7 - 0| = 4/7$. Furthermore, we must have $c_3 = 1$ (otherwise if $c_3 = 0$, $f(x) \geq |x_3 - c_3| = 6/7$) and $c_1 = 1$ (otherwise if $c_1 = 0$, $f(x) \geq |x_1 - c_1| = 5/7$). As a result of (3.41), we must have $f(x) = x_2$ since $x_2 \geq 1 - x_1 \geq 1 - x_3$.
  - $f(x^*) = 3/7$: In this case, we must have $c_2 = 1$, so that $|x_2 - c_2| = |4/7 - 1| = 3/7$. Furthermore, we must have $c_3 = 1$ (otherwise if $c_3 = 0$, $f(x) \geq |x_3 - c_3| = 6/7$) and $c_1 = 1$ (otherwise if $c_1 = 0$, $f(x) \geq |x_1 - c_1| = 5/7$). As a result of (3.41), we must have $f(x) = 1 - x_2$ since since $1 - x_2 \geq 1 - x_1 \geq 1 - x_3$.
  - $f(x^*) = 2/7$ or $f(x^*) = 1/7$: This two cases are impossible because $f(x^*) \geq |x_2 - c_2| = |4/7 - c_2| \geq 3/7$, no matter $c_2 = 0$ or $c_2 = 1$.

While Algorithm 5 is a classical algorithm for querying O_f using a query to $O_{f|D_n}$, it is straightforward to perform this computation in superposition using standard techniques to obtain a quantum query to O_f. However, note that this requires two queries to a quantum oracle for $O_{f|D_n}$ since we must uncompute $f(x^*)$ after computing $f(x)$.

Having the discretization technique at hand, Lemma 3.3 is straightforward.

Proof of Lemma 3.3. Recall that $|\hat{f}(x) - f(x)| \leq \frac{1}{5n}\ \forall\, x \in [0,1]^n$. We run Algorithm 5 to compute $f(x)$ for the queried value of x, except that in Line 3 we take $k^* = \lceil (2n+1)(1-\hat{f}(x^*)) \rfloor$ (here $\lceil a \rfloor$ is the closest integer to a). Because $|\hat{f}(x^*) - f(x^*)| \leq \frac{1}{5n}$, we have

$$\left|(2n+1)(1-\hat{f}(x^*)) - (2n+1)(1-f(x^*))\right| = (2n+1)|\hat{f}(x^*) - f(x^*)| \leq \tfrac{2n+1}{5n} < \tfrac{1}{2}; \quad (3.42)$$

as a result, $k^* = (2n+1)(1-f(x^*))$ because the latter is an integer (see Lemma C.3). Therefore, due to the correctness of Algorithm 5 established in Section C.2, and noticing that the evaluation oracle is only called at Line 3 (with the replacement described above), we successfully simulate one query to O_f by one query to $O_{\hat{f}}$ (actually, to $O_{\hat{f}|D_n}$). □

The full analysis of Algorithm 5 is deferred to Section C.2. In particular,

- In Section C.2.1 we prove that the discretized vector x^* obtained in Line 2 is a good approximation of x in the sense that $\mathrm{Ord}(x^*) = \mathrm{Ord}(x)$;
- In Section C.2.2 we prove that the value k^* obtained in Line 3 satisfies $k^* \in \{1, \ldots, n+1\}$;
- In Section C.2.3 we finally prove that the output returned in Line 4 is correct.

3.3 Proof of Theorem 1.2

We now prove Theorem 1.2 using Theorem 3.2 and Theorem 3.3. Recall that our lower bounds on membership and evaluation queries are both proved on the n-dimensional hypercube. It remains to combine the two lower bounds to establish them simultaneously.

Theorem 3.4. *Let $\mathcal{C}_s := \bigtimes_{i=1}^n [s_i - 2, s_i + 1]$ for some $s \in \{0,1\}^n$. Consider a function $f\colon \mathcal{C}_s \times [0,1]^n \to \mathbb{R}$ such that $f(x) = f_{\mathrm{M}}(x) + f_{\mathrm{E},c}(x)$, where for any $x = (x_1, x_2, \ldots, x_{2n}) \in \mathcal{C}_s \times [0,1]^n$,*

$$f_{\mathrm{M}}(x) = \sum_{i=1}^{n} x_i, \qquad f_{\mathrm{E},c}(x) = \max_{i \in \{n+1,\ldots,2n\}} |x_i - c_{i-n}| \tag{3.43}$$

for some $c \in \{0,1\}^n$. Then outputting an $\tilde{x} \in \mathcal{C}_s \times [0,1]^n$ satisfying

$$f(\tilde{x}) \leq \min_{x \in \mathcal{C}_s \times [0,1]^n} f(x) + \tfrac{1}{3} \tag{3.44}$$

with probability at least 0.8 requires $\Omega(\sqrt{n})$ quantum queries to $O_{\mathcal{C}_s \times [0,1]^n}$ and $\Omega(\sqrt{n}/\log n)$ quantum queries to O_f.

Notice that the dimension of the optimization problem above is $2n$ instead of n; however, the constant overhead of 2 does not influence the asymptotic lower bounds.

Proof of Theorem 3.4. First, we prove that

$$\min_{x \in \mathcal{C}_s \times [0,1]^n} f(x) = S \qquad \text{and} \qquad \arg\min_{x \in \mathcal{C}_s \times [0,1]^n} f(x) = (s - 2_n, c), \tag{3.45}$$

where 2_n is the n-dimensional all-twos vector and $S := \sum_{i=1}^n (s_i - 2)$. On the one hand,

$$f_{\mathrm{M}}(x) \geq S \qquad \forall\, x \in \mathcal{C}_s \times [0,1]^n, \tag{3.46}$$

with equality if and only if $(x_1, \ldots, x_n) = s - 2_n$. On the other hand,

$$f_{\mathrm{E},c}(x) \geq 0 \qquad \forall\, x \in \mathcal{C}_s \times [0,1]^n, \tag{3.47}$$

with equality if and only if $(x_{n+1}, \ldots, x_{2n}) = c$. Thus $f(x) = f_{\mathrm{M}}(x) + f_{\mathrm{E},c}(x) \geq S$ for all $x \in \mathcal{C}_s \times [0,1]^n$, with equality if and only if $x = (x_1, \ldots, x_n, x_{n+1}, \ldots, x_{2n}) = (s - 2_n, c)$.

If we can solve this optimization problem with an output $\tilde{x}$ satisfying (3.44), then

$$f_{\mathrm{M}}(\tilde{x}) + f_{\mathrm{E},c}(\tilde{x}) = f(\tilde{x}) \leq S + \tfrac{1}{3}. \tag{3.48}$$

Eqs. (3.46), (3.47), and (3.48) imply

$$f_{\mathrm{M}}(\tilde{x}) \leq S + \tfrac{1}{3} = \min_{x \in \mathcal{C}_s \times [0,1]^n} f_{\mathrm{M}}(x) + \tfrac{1}{3}; \tag{3.49}$$

$$f_{\mathrm{E},c}(\tilde{x}) \leq \tfrac{1}{3} = \min_{x \in \mathcal{C}_s \times [0,1]^n} f_{\mathrm{E},c}(x) + \tfrac{1}{3}. \tag{3.50}$$

On the one hand, Eq. (3.49) says that $\tilde{x}$ also minimizes f_{M} with approximation error $\epsilon = \frac{1}{3}$. By Theorem 3.2, this requires $\Omega(\sqrt{n})$ queries to the membership oracle $O_{\mathcal{C}_s}$. Also notice that one query to $O_{\mathcal{C}_s \times [0,1]^n}$ can be trivially simulated one query to $O_{\mathcal{C}_s}$; therefore,

minimizing f with approximation error $\epsilon = \frac{1}{3}$ with success probability 0.9 requires $\Omega(\sqrt{n})$ quantum queries to $O_{\mathcal{C}_s \times [0,1]^n}$.

On the other hand, Eq. (3.50) says that $\tilde{x}$ minimizes $f_{\mathrm{E},c}$ with approximation error $\epsilon = \frac{1}{3}$. By Theorem 3.3, it takes $\Omega(\sqrt{n}/\log n)$ queries to $O_{f_{\mathrm{E},c}}$ to output $\tilde{x}$. Also notice that

$$f(x) = f_{\mathrm{M}}(x) + f_{\mathrm{E},c}(x) = \sum_{i=1}^{n} x_i + f_{\mathrm{E},c}(x); \tag{3.51}$$

therefore, one query to O_f can be simulated by one query to $O_{f_{\mathrm{E},c}}$. Therefore, approximately minimizing f with success probability 0.9 requires $\Omega(\sqrt{n}/\log n)$ quantum queries to O_f.

In addition, f_{M} is independent of the coordinates $x_{n+1}, \ldots, x_{2n}$ and only depends on the coordinates $x_1, \ldots, x_n$, whereas $f_{\mathrm{E},c}$ is independent of the coordinates $x_1, \ldots, x_n$ and only depends on the coordinates $x_{n+1}, \ldots, x_{2n}$. As a result, the oracle $O_{\mathcal{C}_s \times [0,1]^n}$ reveals no information about c, and O_f reveals no information about s. Since solving the optimization problem reveals both s and c, the lower bounds on query complexity must hold simultaneously.

Overall, to output an $\tilde{x} \in \mathcal{C}_s \times [0,1]^n$ satisfying (3.44) with success probability at least $0.9 \cdot 0.9 > 0.8$, we need $\Omega(\sqrt{n})$ quantum queries to $O_{\mathcal{C}_s \times [0,1]^n}$ and $\Omega(\sqrt{n}/\log n)$ quantum queries to O_f, as claimed. □

3.4 Smoothed hypercube

As a side point, our quantum lower bound in Theorem 3.4 also holds for a smooth convex body. Given an n-dimensional hypercube $\mathcal{C}_{x,l} := \bigtimes_{i=1}^{n} [x_i - l, x_i]$, we define a smoothed version as

$$\mathcal{SC}_{x,l} := B_2\Big(\bigtimes_{i=1}^{n} \Big[x_i - \frac{2n}{2n+1} l, x_i - \frac{1}{2n+1} l \Big], \frac{1}{2n+1} l \Big) \tag{3.52}$$

using Definition 2.3. For instance, a smoothed 3-dimensional cube is shown in Figure 1.

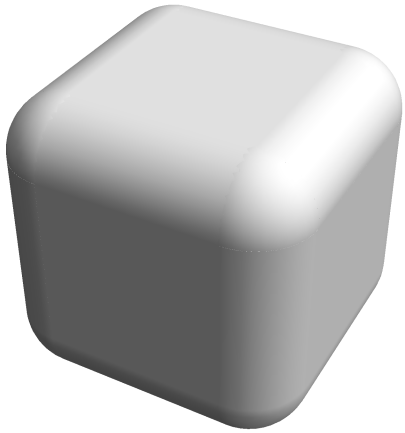

Figure 1: Smoothed hypercube of dimension 3.

The smoothed hypercube satisfies

$$\mathcal{C}_{x-\frac{1}{2n+1}l_n,\frac{2n-1}{2n+1}l} \subseteq \mathcal{SC}_{x,l} \subseteq \mathcal{C}_{x,l} \tag{3.53}$$

where l_n is l times the n-dimensional all-ones vector; in other words, it is contained in the original (non-smoothed) hypercube, and it contains the hypercube with the same center but edge length $\frac{2n-1}{2n+1}l$. For instance, $\bigtimes_{i=1}^n [\frac{1}{2n+1}, \frac{2n}{2n+1}] \subseteq \mathcal{SC}_{1_n,1} \subseteq \bigtimes_{i=1}^n [0,1]$; by Eq. (3.34), $D_n \subseteq \mathcal{SC}_{1_n,1}$. It can be verified that the proof of Theorem 3.2 still holds if the hypercube $\bigtimes_{i=1}^n [s_i - 2, s_i + 1] = \mathcal{C}_{s+1_n,3}$ is replaced by $\mathcal{SC}_{s+1_n,3}$, and the proof of Theorem 3.3 still holds if the unit hypercube $[0,1]^n$ is replaced by $\mathcal{SC}_{1_n,1}$; consequently Theorem 3.4 also holds. More generally, the proofs remain valid as long as the smoothed hypercube is contained in $[0,1]^n$ and contains D_n (for discretization).

Acknowledgements

We thank Yin Tat Lee for numerous helpful discussions and anonymous reviewers for suggestions on preliminary versions of this paper. We also thank Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf for sharing a preliminary version of their manuscript [4] and for their detailed feedback on a preliminary version of this paper, including identifying some mistakes in previous lower bound arguments and pointing out a minor technical issue in the evaluation of the height function in Line 6 of Algorithm 4 (and in [21]). This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Algorithms Teams program. AMC also received support from the Army Research Office (MURI award W911NF-16-1-0349), the Canadian Institute for Advanced Research, and the National Science Foundation (grant CCF-1526380). XW also received support from the National Science Foundation (grants CCF-1755800 and CCF-1816695).

References

[1] Andris Ambainis and Ashley Montanaro, *Quantum algorithms for search with wildcards and combinatorial group testing*, Quantum Information and Computation **14** (2014), no. 5&6, 439–453, arXiv:1210.1148. https://doi.org/10.26421/QIC14.5-6

[2] Joran van Apeldoorn and András Gilyén, *Improvements in quantum SDP-solving with applications*, Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, Leibniz International Proceedings in Informatics (LIPIcs), vol. 132, pp. 99:1–99:15, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, arXiv:1804.05058. https://doi.org/10.4230/LIPIcs.ICALP.2019.99

[3] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf, *Quantum SDP-solvers: Better upper and lower bounds*, Proceedings of the 58th Annual Symposium on Foundations of Computer Science, 2017, arXiv:1705.01843. https://doi.org/10.1109/FOCS.2017.44

[4] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf, *Convex optimization using quantum oracles*, Quantum, 2019, arXiv:1809.00643.

[5] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge University Press, 2004.

[6] Fernando G.S.L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu, *Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning*, Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, Leibniz International Proceedings in Informatics (LIPIcs), vol. 132, pp. 27:1–27:14, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, arXiv:1710.02581v2. https://doi.org/10.4230/LIPIcs.ICALP.2019.27

[7] Fernando G.S.L. Brandão and Krysta Svore, *Quantum speed-ups for solving semidefinite programs*, Proceedings of the 58th Annual Symposium on Foundations of Computer Science, pp. 415–426, 2017, arXiv:1609.05537. https://doi.org/10.1109/FOCS.2017.45

[8] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp, *Quantum amplitude amplification and estimation*, Contemporary Mathematics **305** (2002), 53–74, arXiv:quant-ph/0005055.

[9] Andrew M. Childs, Robin Kothari, and Rolando D. Somma, *Quantum algorithm for systems of linear equations with exponentially improved dependence on precision*, SIAM Journal on Computing **46** (2017), no. 6, 1920–1950, arXiv:1511.02306. https://doi.org/10.1137/16M1087072

[10] George B. Dantzig and Mukund N. Thapa, *Linear programming 2: Theory and extensions*, Springer, 2006.

[11] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe, *Optimizing quantum optimization algorithms via faster quantum gradient computation*, Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1425–1444, Society for Industrial and Applied Mathematics, 2019, arXiv:1711.00465. https://doi.org/10.1137/1.9781611975482.87

[12] Martin Grötschel, László Lovász, and Alexander Schrijver, *Geometric algorithms and combinatorial optimization*, Algorithms and Combinatorics, vol. 2, Springer, 2012.

[13] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd, *Quantum algorithm for linear systems of equations*, Physical Review Letters **103** (2009), no. 15, 150502, arXiv:0811.3171. https://doi.org/10.1103/PhysRevLett.103.150502

[14] Lars Hörmander, *The analysis of linear partial differential operators: Distribution theory and Fourier analysis*, Springer-Verlag, 1990.

[15] Stephen P. Jordan, *Fast quantum algorithm for numerical gradient estimation*, Physical Review Letters **95** (2005), no. 5, 050501, arXiv:quant-ph/0405146. https://doi.org/10.1103/PhysRevLett.95.050501

[16] Adam T. Kalai and Santosh Vempala, *Simulated annealing for convex optimization*, Mathematics of Operations Research **31** (2006), no. 2, 253–266. https://doi.org/10.1287/moor.1060.0194

[17] Narendra Karmarkar, *A new polynomial-time algorithm for linear programming*, Proceedings of the 16th Annual ACM Symposium on Theory of Computing, pp. 302–311, 1984. https://doi.org/10.1145/800057.808695

[18] James E. Kelley, Jr., *The cutting-plane method for solving convex programs*, Journal of the Society for Industrial and Applied Mathematics **8** (1960), no. 4, 703–712.

[19] Iordanis Kerenidis and Anupam Prakash, *Quantum gradient descent for linear systems and least squares*, 2017, arXiv:1704.04992.

[20] Yin Tat Lee, *personal communication*, 2018.

[21] Yin Tat Lee, Aaron Sidford, and Santosh S. Vempala, *Efficient convex optimization with*

membership oracles, Proceedings of the 31st Conference on Learning Theory, Proceedings of Machine Learning Research, vol. 75, pp. 1292–1294, 2018, arXiv:1706.07357.

[22] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong, *A faster cutting plane method and its implications for combinatorial and convex optimization*, Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science, pp. 1049–1065, 2015, arXiv:1508.04874. https://doi.org/10.1109/FOCS.2015.68

[23] László Lovász and Santosh Vempala, *Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization*, Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 57–68, 2006. https://doi.org/10.1109/FOCS.2006.28

[24] Arkadi S. Nemirovski, *Information-based complexity of convex programming*, lecture notes, 1995.

[25] Arkadi S. Nemirovski and David B. Yudin, *Problem complexity and method efficiency in optimization*, Wiley, 1983.

[26] Yurii Nesterov, *Introductory lectures on convex optimization: A basic course*, Applied Optimization, vol. 87, Springer, 2013.

[27] Patrick Rebentrost, Maria Schuld, Leonard Wossnig, Francesco Petruccione, and Seth Lloyd, *Quantum gradient descent and Newton's method for constrained polynomial optimization*, New Journal of Physics **21** (2019), no. 7, 073023, IOP Publishing, arXiv:1612.01789. https://doi.org/10.1088/1367-2630/ab2a9e

[28] Pravin M. Vaidya, *A new algorithm for minimizing convex functions over convex sets*, Proceedings of the 30th Annual Symposium on Foundations of Computer Science, pp. 338–343, 1989. https://doi.org/10.1109/SFCS.1989.63500

A Auxiliary lemmas

A.1 Classical gradient computation

Here we prove that the classical query complexity of gradient computation is linear in the dimension.

Lemma A.1. *Let f be an L-Lipschitz convex function that is specified by an evaluation oracle with precision $\delta = 1/\operatorname{poly}(n)$. Any (deterministic or randomized) classical algorithm to calculate a subgradient of f with L_∞-norm error $\epsilon = 1/\operatorname{poly}(n)$ must make $\tilde{\Omega}(n)$ queries to the evaluation oracle.*

Proof. Consider the linear function $f(x) = c^T x$ where each $c_i \in [0, 1]$. Since each c_i must be determined to precision ϵ, the problem hides $n \log(1/\epsilon)$ bits of information. Furthermore, since the evaluation oracle has precision δ, each query reveals only $\log(1/\delta)$ bits of information. Thus any classical algorithm must make at least $\frac{n \log(1/\epsilon)}{\log(1/\delta)} = n/\log(n)$ evaluation queries. □

A.2 Mollified functions

The following lemma establishes properties of mollified functions:

Lemma A.2 (Mollifier properties)**.** *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz convex function with mollification $F_\delta = f * m_\delta$, where m_δ is defined in (2.11). Then*

(i) *F_δ is infinitely differentiable,*
(ii) *F_δ is convex,*
(iii) *F_δ is L-Lipschitz continuous, and*
(iv) *$|F_\delta(x) - f(x)| \leq L\delta$.*

Proof.

(i) Convolution satisfies $\frac{\mathrm{d}(p*q)}{\mathrm{d}x} = p * \frac{\mathrm{d}q}{\mathrm{d}x}$, so because m_δ is infinitely differentiable, F_δ is infinitely differentiable.

(ii) We have $F_\delta(x) = \int_{\mathbb{R}^n} f(x-z)m_\delta(z)\,\mathrm{d}z = \int_{\mathbb{R}^n} f(z)m_\delta(x-z)\,\mathrm{d}z$. Thus

$$F_\delta(\lambda x + (1-\lambda)y) = \int_{\mathbb{R}^n} f(\lambda x + (1-\lambda)y - z)m_\delta(z)\,\mathrm{d}z \tag{A.1}$$

$$\geq \int_{\mathbb{R}^n} [\lambda f(x-z) + (1-\lambda)f(y-z)]m_\delta(z)\,\mathrm{d}z \tag{A.2}$$

$$= \lambda F_\delta(x) + (1-\lambda)F_\delta(y), \tag{A.3}$$

where the inequality holds by convexity of f and the fact that $m_\delta \geq 0$. Thus F_δ is convex.

(iii) We have

$$\|F_\delta(x) - F_\delta(y)\| = \left\| \int_{\mathbb{R}^n} [f(x-z) - f(y-z)]m_\delta(z)\,\mathrm{d}z \right\| \tag{A.4}$$

$$\leq \int_{\mathbb{R}^n} \|f(x-z) - f(y-z)\|m_\delta(z)\,\mathrm{d}z \tag{A.5}$$

$$\leq L\|x-y\| \int_{\mathbb{R}^n} m_\delta(z)\,\mathrm{d}z \tag{A.6}$$

$$= L\|x-y\|. \tag{A.7}$$

Thus from Definition 2.9, F_δ is L-Lipschitz.

(iv) We have

$$|F_\delta(x) - f(x)| = \left| \int_{\mathbb{R}^n} f(x-z)g(z)\,\mathrm{d}z - \int_{\mathbb{R}^n} f(x)g(z)\,\mathrm{d}z \right| \tag{A.8}$$

$$\leq \int_{\mathbb{R}^n} |f(x-z) - f(z)|\, g(z)\,\mathrm{d}z \tag{A.9}$$

$$\leq L \int_{\mathbb{R}^n} |z|\, g(z)\,\mathrm{d}z \tag{A.10}$$

$$= L \int_{B_2(0,\delta)} \frac{|z|}{I_n} \exp\left(-\frac{1}{1-\|z/\delta\|^2}\right) \mathrm{d}z \tag{A.11}$$

$$= L\delta \int_{B_2(0,1)} \frac{|u|}{I_n} \exp\left(-\frac{1}{1-\|u\|^2}\right) \mathrm{d}u \tag{A.12}$$

$$\leq L\delta \int_{B_2(0,1)} \frac{1}{I_n} \exp\left(-\frac{1}{1-\|u\|^2}\right) \mathrm{d}u \tag{A.13}$$

$$= L\delta \tag{A.14}$$

as claimed.

□

The following lemma shows strong convexity of mollified functions, ruling out the possibility of directly applying Lemma B.2 to calculate subgradients.

Lemma A.3. *There exists a 1-Lipschitz convex function f such that for any β-smooth function g with $|f(x) - g(x)| \leq \delta$ for all x, $\beta\delta \geq c$ where c is a constant.*

Proof. Let $f(x) = |x|$. Consider $x \geq 0$. By the smoothness of g,

$$g(x) \leq g(0) + \nabla g(0)^T x + \frac{\beta}{2}x^2, \tag{A.15}$$

$$g(-x) \leq g(0) - \nabla g(0)^T x + \frac{\beta}{2}x^2. \tag{A.16}$$

As a result, we have $g(x) + g(-x) \leq 2g(0) + \beta x^2$ for all $x > 0$. Since $|f(x) - g(x)| \leq \delta$,

$$f(x) + f(-x) \leq 2f(0) + \beta x^2 + 4\delta \quad \Rightarrow \quad \beta x^2 - 2x + 4\delta \geq 0 \tag{A.17}$$

for all $x > 0$.

Since $4\delta > 0$, the discriminant must be non-positive. Therefore, $16 - 16\beta\delta \leq 0$, so $\beta\delta \geq 1$. □

B Proof details for upper bound

We give the complete proof of Lemma 2.2 in this section.

Given a quantum oracle that computes the function N_0F in the form

$$U_F |x\rangle |y\rangle = |x\rangle |y \oplus (N_0F(x) \mod N)\rangle , \tag{B.1}$$

it is well known that querying U_F with

$$|y_0\rangle = \frac{1}{\sqrt{N_0}} \sum_{i\in\{0,1,\ldots,N-1\}} e^{\frac{2\pi ix}{N_0}} |i\rangle \tag{B.2}$$

allows us to implement the phase oracle O_F in one query. This is a common technique used in quantum algorithms known as *phase kickback*.

First, we prove the following lemma:

Lemma B.1. *Let $G := \{-N/2, -N/2+1, \ldots, N/2-1\}$ and define $\gamma\colon \{0,1,\ldots,N-1\} \to G$ by $\gamma(x) = x - N/2$ for all $x \in \{0,1,\ldots,N-1\}$. Consider the inverse quantum Fourier transforms*

$$\mathrm{QFT}_N^{-1} |x\rangle := \frac{1}{\sqrt{N}} \sum_{y\in[0,N-1]} e^{-\frac{2\pi ixy}{N}} |y\rangle , \quad \forall\, x \in [0, N-1]; \tag{B.3}$$

$$\mathrm{QFT}_G^{-1} |\gamma(x)\rangle := \frac{1}{\sqrt{N}} \sum_{\gamma(y)\in G} e^{-\frac{2\pi i\gamma(x)\gamma(y)}{N}} |\gamma(y)\rangle , \quad \forall\, \gamma(x) \in G \tag{B.4}$$

over $[0, N-1] := \{0,1,\ldots,N-1\}$ and G, respectively. Then we have $\mathrm{QFT}_G^{-1} = U\,\mathrm{QFT}_N^{-1}\,U$, where U is a tensor product of $b = \log_2 N$ single-qubit unitaries.

Proof. For any $x \in [0, N-1]$, we have

$$\mathrm{QFT}_G^{-1} |x\rangle = \frac{1}{\sqrt{N}} \sum_{y\in[0,N-1]} e^{-\frac{2\pi i\gamma(x)\gamma(y)}{N}} |y\rangle \tag{B.5}$$

which is equivalent to

$$\frac{1}{\sqrt{N}} \sum_{y\in[0,N-1]} e^{-\frac{2\pi ixy}{N}} e^{\pi i(x+y)} |y\rangle \tag{B.6}$$

up to a global phase. Setting $U|x\rangle = e^{\pi ix}|x\rangle$ for all $x \in \{0,1,\ldots,N-1\}$, we have the result. □

The above shows that we can implement QFT_G^{-1} on a single b-bit register using $O(b)$ gates. Thus there is no significant overhead in gate complexity that results from using QFT_G instead of the usual QFT.

Now we prove Lemma 2.2, which is rewritten below:

Lemma B.2. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz function that is specified by an evaluation oracle with error at most ϵ. Let f be β-smooth in $B_\infty(x, 2\sqrt{\epsilon/\beta})$, and let $\tilde{g}$ be the output of* `GradientEstimate`$(f, \epsilon, L, \beta, x_0)$ *(from Algorithm 1). Let $g = \nabla f(x_0)$. Then*

$$\Pr\left[|\tilde{g}_i - g_i| > 1500\sqrt{n\epsilon\beta}\right] < \frac{1}{3}, \quad \forall\, i \in [n]. \tag{B.7}$$

Proof. To analyze the `GradientEstimate` algorithm, let the actual state obtained before applying the inverse QFT over G be

$$|\psi\rangle = \frac{1}{N^{n/2}} \sum_{x \in G^d} e^{2\pi i \tilde{F}(x)} |x\rangle , \tag{B.8}$$

where $|\tilde{F}(x) - \frac{N}{2Ll}[f(x_0 + \frac{lx}{N}) - f(x_0)]| \leq \frac{1}{N_0}$. Also consider the idealized state

$$|\phi\rangle = \frac{1}{N^{n/2}} \sum_{x \in G^d} e^{\frac{2\pi i g \cdot x}{2L}} |x\rangle . \tag{B.9}$$

From Lemma B.1 we can efficiently apply the inverse QFT over G; from the analysis of phase estimation (see [8]), we know that

$$\forall\, i \in [n] \quad \Pr\left[\left|\frac{Ng_i}{2L} - k_i\right| > w\right] < \frac{1}{2(w-1)}, \tag{B.10}$$

so in particular,

$$\forall\, i \in [n] \quad \Pr\left[\left|\frac{Ng_i}{2L} - k_i\right| > 4\right] < \frac{1}{6}. \tag{B.11}$$

Now, let $g = \nabla f(x_0)$. The difference in the probabilities of any measurement on $|\psi\rangle$ and $|\phi\rangle$ is bounded by the trace distance between the two density matrices, which is

$$\||\psi\rangle\langle\psi| - |\phi\rangle\langle\phi|\|_1 = 2\sqrt{1 - |\langle\psi|\phi\rangle|^2} \leq 2\||\psi\rangle - |\phi\rangle\|. \tag{B.12}$$

Since f is β-smooth in $B_\infty(x, 2\sqrt{\frac{\epsilon}{\beta}})$,

$$\begin{aligned}
\tilde{F}(x) &\leq \frac{N}{2Ll}\left[f\left(x_0 + \frac{lx}{N}\right) - f(x_0)\right] + \frac{1}{N_0} && \text{(B.13)}\\
&\leq \frac{N}{2Ll}\left(\frac{l}{N}\nabla f(x_0)\cdot x + \frac{\beta l^2 x^2}{2N^2}\right) + \frac{1}{N_0} && \text{(B.14)}\\
&\leq \frac{1}{2L}\nabla f(x_0)\cdot x + \frac{N\beta l n}{4L} + \frac{N\epsilon}{Ll}. && \text{(B.15)}
\end{aligned}$$

Then we have

$$\begin{aligned}
\||\psi\rangle - |\phi\rangle\|^2 &= \frac{1}{N^d}\sum_{x\in G_d} |e^{2\pi i\tilde{F}(x)} - e^{\frac{2\pi i g.x}{2L}}|^2 && \text{(B.16)}\\
&= \frac{1}{N^d}\sum_{x\in G_d} 4\pi^2\left(\tilde{F}(x) - \frac{g\cdot x}{2L}\right)^2 && \text{(B.17)}\\
&\leq \frac{1}{N^d}\sum_{x\in G_d} \frac{4\pi^2 N^2}{L^2}\left(\frac{\beta l n}{4} + \frac{\epsilon}{l}\right)^2 && \text{(B.18)}\\
&= \frac{4\pi^2 N^2 \beta\epsilon n}{L^2}. && \text{(B.19)}
\end{aligned}$$

In Algorithm 1, N is chosen such that $N \le \frac{L}{24\pi\sqrt{n\epsilon\beta}}$. Plugging this into (B.16),

$$\| |\psi\rangle - |\phi\rangle \|^2 \le \frac{1}{144}. \tag{B.20}$$

Thus the trace distance is at most $\frac{1}{6}$. Therefore, $\Pr\left[|k_i - \frac{Ng_i}{2L}| > 4\right] < \frac{1}{3}$. Thus we have

$$\Pr\left[|\tilde{g}_i - g_i| > \frac{8L}{N}\right] < \frac{1}{3}, \quad \forall i \in [n]. \tag{B.21}$$

From Algorithm 1, $\frac{1}{N} \le \frac{48\pi\sqrt{n\epsilon\beta}}{L}$, so $\frac{8L}{N} < 384\pi\sqrt{n\epsilon\beta} < 1500\sqrt{n\epsilon\beta}$, and the result follows. □

Finally, we prove that the height function h_p can be evaluated with precision ϵ using $O(\log(1/\epsilon))$ queries to a membership oracle:

Lemma B.3. *The function $h_p(x)$ can be evaluated for any $x \in B_\infty(0, r/2)$ with any precision $\epsilon \ge 7\kappa\delta$ using $O(\log(1/\epsilon))$ queries to a membership oracle with error δ.*

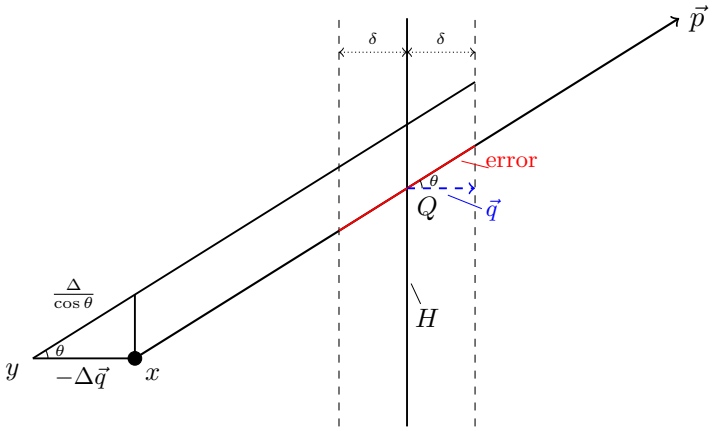

Figure 2: Relating the error to 2δ in $n = 2$ dimensions.

Proof. We denote the intersection of the ray $x + t\vec{p}$ and the boundary of K by Q, and let H be an $(n-1)$-dimensional hyperplane that is tangent to K at Q. Because K is convex, it lies on only one side of H; we let $\vec{q}$ denote the unit vector at Q that is perpendicular to H and points out of K. Let $\theta := \arccos\langle \vec{p}, \vec{q} \rangle$.

Using binary search with $\log(1/\delta)$ queries, we can find a point P on the ray $x + t\vec{p}$ such that $P \notin B(K, -\delta)$ and $P \in B(K, \delta)$. The total error for t is then at most $\frac{2\delta}{\cos\theta}$. Now consider $y = x - \Delta\vec{q}$ for some small $\Delta > 0$. Then $h_p(y) - h_p(x) = \frac{\Delta}{\cos\theta} + o(\frac{\Delta}{\cos\theta})$ (see Figure 2 for an illustration with $n = 2$).

By Proposition 2.2, $h_p(x)$ is 3κ-Lipschitz for any $x \in B(0, r/2)$; therefore, $h_p(y) - h_p(x) \le 3\kappa\|y - x\| = 3\kappa\Delta$, and hence

$$\frac{\Delta}{\cos\theta} + o\Big(\frac{\Delta}{\cos\theta}\Big) \le 3\kappa\Delta \quad \Rightarrow \quad \frac{1}{\cos\theta} \le 3.5\kappa \tag{B.22}$$

for a small enough $\Delta > 0$. Thus the error in $h_p(x)$ is at most $\frac{2\delta}{\cos\theta} \le 7\kappa\delta$, and the result follows. □

C Proof details for lower bound

In this section, we give proof details for our claims in Section 3.2.

C.1 Convexity of max-norm optimization

In this subsection, we prove:

Lemma C.1. *The function*

$$f_c(x) = \max_{i\in[n]} |\pi(x_i) - c_i| + \Big(\sum_{i=1}^{n} |\pi(x_i) - x_i|\Big) \tag{C.1}$$

is convex on $\mathbb{R}^n$*, where* $c \in \{0,1\}^n$ *and* $\pi\colon \mathbb{R} \to [0,1]$ *is defined as*

$$\pi(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \le x \le 1 \\ 1 & \text{if } x > 1. \end{cases} \tag{C.2}$$

Proof. For convenience, we define $g_i\colon \mathbb{R}^n \to \mathbb{R}$ for $i \in [n]$ as

$$g_i(x) := |\pi(x_i) - x_i| = \begin{cases} -x_i & \text{if } x_i < 0 \\ 0 & \text{if } 0 \le x_i \le 1 \\ x_i - 1 & \text{if } x_i > 1 \end{cases} \tag{C.3}$$

where the second equality follows from (C.2). It is clear that $g_i(x) = \max\{-x_i, 0, x_i - 1\}$ by (C.3). Since the pointwise maximum of convex functions is convex, $g_i(x)$ is convex for all $i \in [n]$.

Moreover, for all $i \in [n]$ we define $h_{c,i}\colon \mathbb{R}^n \to \mathbb{R}$ as $h_{c,i}(x) := |\pi(x_i) - c_i| + |\pi(x_i) - x_i|$. We claim that $h_{c,i}(x) = |x_i - c_i|$, and thus $h_{c,i}$ is convex. If $c_i = 0$, then $|\pi(x_i) - c_i| + |\pi(x_i) - x_i| = \pi(x_i) + |\pi(x_i) - x_i|$; as a result,

$$x_i < 0 \quad \Rightarrow \quad \pi(x_i) + |\pi(x_i) - x_i| = 0 + |0 - x_i| = -x_i; \tag{C.4}$$

$$0 \le x_i \le 1 \quad \Rightarrow \quad \pi(x_i) + |\pi(x_i) - x_i| = x_i + |x_i - x_i| = x_i; \tag{C.5}$$

$$x_i > 1 \quad \Rightarrow \quad \pi(x_i) + |\pi(x_i) - x_i| = 1 + |1 - x_i| = x_i. \tag{C.6}$$

Therefore, $\forall\, i \in [n], h_{c,i}(x) = |x_i - c_i|$. The proof is similar when $c_i = 1$.

Now we have

$$f_c(x) = \max_{i\in[n]} \Big(|\pi(x_i) - c_i| + \sum_{j=1}^{n} |\pi(x_j) - x_j|\Big) \tag{C.7}$$

$$= \max_{i\in[n]} \Big((|\pi(x_i) - c_i| + |\pi(x_i) - x_i|) + \sum_{j\neq i} g_j(x)\Big) \tag{C.8}$$

$$= \max_{i\in[n]} \Big(h_{c,i}(x) + \sum_{j\neq i} g_j(x)\Big). \tag{C.9}$$

Because $h_{c,i}$ and g_j are both convex functions on $\mathbb{R}^n$ for all $i, j \in [n]$, the function $h_{c,i}(x) + \sum_{j\neq i} g_j(x)$ is convex on $\mathbb{R}^n$. Thus f_c is the pointwise maximum of n convex functions and is therefore itself convex. □

C.2 Proof of Lemma 3.4

C.2.1 Correctness of Line 1 and Line 2

In this subsection, we prove:

Lemma C.2. *Let b and σ be the values computed in Line 1 of Algorithm 5, and let $x^* = \chi(b, \sigma^{-1})$. Then $\mathrm{Ord}(x^*) = \mathrm{Ord}(x)$.*

Proof. First, observe that $b \in \{0,1\}^n$ and $\sigma \in S_n$ because

- For all $i \in [n]$, both x_i and $1 - x_i$ can be written as $b_i x_i + (1 - b_i)(1 - x_i)$ for some $b_i \in \{0, 1\}$;
- $\mathrm{Ord}(x)$ is *palindrome*, i.e., if x_{i_1} is the largest in $\{x_1, \ldots, x_n, 1 - x_1, \ldots, 1 - x_n\}$ then $1 - x_{i_1}$ is the smallest in $\{x_1, \ldots, x_n, 1 - x_1, \ldots, 1 - x_n\}$; if $1 - x_{i_2}$ is the second largest in $\{x_1, \ldots, x_n, 1 - x_1, \ldots, 1 - x_n\}$ then x_{i_2} is the second smallest in $\{x_1, \ldots, x_n, 1 - x_1, \ldots, 1 - x_n\}$; etc.

Recall that in (3.38), the decreasing order of $\{x_1, \ldots, x_n, 1 - x_1, \ldots, 1 - x_n\}$ is

$$\begin{aligned} &b_{\sigma(1)}x_{\sigma(1)} + (1 - b_{\sigma(1)})(1 - x_{\sigma(1)}) \geq \cdots \geq b_{\sigma(n)}x_{\sigma(n)} + (1 - b_{\sigma(n)})(1 - x_{\sigma(n)}) \\ &\quad \geq (1 - b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1 - x_{\sigma(n)}) \geq \cdots \geq (1 - b_{\sigma(1)})x_{\sigma(1)} + b_{\sigma(1)}(1 - x_{\sigma(1)}). \end{aligned} \tag{C.10}$$

On the other hand, by the definition of D_n, we have

$$\{x_1^*, \ldots, x_n^*, 1 - x_1^*, \ldots, 1 - x_n^*\} = \Big\{\frac{1}{2n+1}, \frac{2}{2n+1}, \ldots, \frac{2n}{2n+1}\Big\}. \tag{C.11}$$

Combining (C.10) and (C.11), it suffices to prove that for any $i \in [n]$,

$$b_{\sigma(i)}x^*_{\sigma(i)} + (1 - b_{\sigma(i)})(1 - x^*_{\sigma(i)}) = 1 - \frac{i}{2n+1}; \tag{C.12}$$

$$(1 - b_{\sigma(i)})x^*_{\sigma(i)} + b_{\sigma(i)}(1 - x^*_{\sigma(i)}) = \frac{i}{2n+1}. \tag{C.13}$$

We only prove (C.12); the proof of (C.13) follows symmetrically.

By (3.35), we have $x_j^* = (1 - b_j)\frac{\sigma^{-1}(j)}{2n+1} + b_j(1 - \frac{\sigma^{-1}(j)}{2n+1})$ for all $j \in [n]$; taking $j = \sigma(i)$, we have $x^*_{\sigma(i)} = (1 - b_{\sigma(i)})\frac{i}{2n+1} + b_{\sigma(i)}(1 - \frac{i}{2n+1})$. Moreover, since $b_{\sigma(i)} \in \{0, 1\}$ implies that $b_{\sigma(i)}(1 - b_{\sigma(i)}) = 0$ and $b^2_{\sigma(i)} + (1 - b_{\sigma(i)})^2 = 1$, we have

$$\begin{aligned} &b_{\sigma(i)}x^*_{\sigma(i)} + (1 - b_{\sigma(i)})(1 - x^*_{\sigma(i)}) \\ &\quad = b_{\sigma(i)}\big[(1 - b_{\sigma(i)})\tfrac{i}{2n+1} + b_{\sigma(i)}(1 - \tfrac{i}{2n+1})\big] \\ &\qquad + (1 - b_{\sigma(i)})\big[b_{\sigma(i)}\tfrac{i}{2n+1} + (1 - b_{\sigma(i)})(1 - \tfrac{i}{2n+1})\big] && \text{(C.14)} \\ &\quad = 2b_{\sigma(i)}(1 - b_{\sigma(i)})\tfrac{i}{2n+1} + \big(b^2_{\sigma(i)} + (1 - b_{\sigma(i)})^2\big)(1 - \tfrac{i}{2n+1}) && \text{(C.15)} \\ &\quad = 1 - \tfrac{i}{2n+1}, && \text{(C.16)} \end{aligned}$$

which is exactly (C.12). □

C.2.2 Correctness of Line 3

In this subsection, we prove:

Lemma C.3. *There is some $k^* \in \{1, \ldots, n+1\}$ such that $f(x^*) = 1 - \frac{k^*}{2n+1}$.*

Proof. Because $|x_i^* - c_i|$ is an integer multiple of $\frac{1}{2n+1}$ for all $i \in [n]$, $f(x^*)$ must also be an integer multiple of $\frac{1}{2n+1}$. As a result, $k^* = (2n+1)(1 - f(x^*)) \in \mathbb{Z}$.

It remains to prove that $1 \le k^* \le n+1$. By the definition of D_n in (3.34), we have

$$x_i^* = (1 - b_i)\frac{\sigma^{-1}(i)}{2n+1} + b_i\Big(1 - \frac{\sigma^{-1}(i)}{2n+1}\Big) \qquad \forall\, i \in [n]; \tag{C.17}$$

since $b_i = 0$ or 1, we have $x_i^* \in \{\frac{\sigma^{-1}(i)}{2n+1}, 1 - \frac{\sigma^{-1}(i)}{2n+1}\}$. Because we also have $c_i \in \{0, 1\}$,

$$|x_i^* - c_i| \le 1 - \frac{\sigma^{-1}(i)}{2n+1} \le \frac{2n}{2n+1}. \tag{C.18}$$

As a result,

$$f(x^*) = \max_{i \in [n]} |x_i^* - c_i| \le \frac{2n}{2n+1} \;\Rightarrow\; k^* \ge 1. \tag{C.19}$$

It remains to prove $k^* \le n+1$. By (C.17), we have

$$x_{\sigma(n)}^* \in \Big\{\frac{n}{2n+1}, \frac{n+1}{2n+1}\Big\}; \tag{C.20}$$

because $c_{\sigma(n)} \in \{0, 1\}$, we have

$$|x_{\sigma(n)}^* - c_{\sigma(n)}| \ge \frac{n}{2n+1}. \tag{C.21}$$

Therefore, we have

$$f(x^*) = \max_{i \in [n]} |x_i^* - c_i| \ge |x_{\sigma(n)}^* - c_{\sigma(n)}| \ge \frac{n}{2n+1}, \tag{C.22}$$

which implies $k^* \le n+1$. □

C.2.3 Correctness of Line 4

In this subsection, we prove:

Lemma C.4. *The output of $f(x)$ in Line 4 is correct.*

Proof. A key observation we use in the proof, following directly from (C.17), is that

$$|x_{\sigma(i)}^* - c_{\sigma(i)}| = \begin{cases} \frac{i}{2n+1} & \text{if } c_{\sigma(i)} = b_{\sigma(i)}; \\ 1 - \frac{i}{2n+1} & \text{if } c_{\sigma(i)} = 1 - b_{\sigma(i)}. \end{cases} \tag{C.23}$$

First, assume that $k^* \in \{1, \dots, n\}$ (i.e., the "otherwise" case in (3.39) happens). By (C.23),

$$x^*_{\sigma(k^*)} \in \Big\{\frac{k^*}{2n+1}, 1 - \frac{k^*}{2n+1}\Big\}; \quad x^*_{\sigma(i)} \notin \Big\{\frac{k^*}{2n+1}, 1 - \frac{k^*}{2n+1}\Big\} \quad \forall\, i \neq k^*, \tag{C.24}$$

which implies that for all $i \neq k^*$, $|x^*_{\sigma(i)} - c_{\sigma(i)}| \neq 1 - \frac{k^*}{2n+1}$ since $c_{\sigma(i)} \in \{0, 1\}$. As a result, we must have

$$|x^*_{\sigma(k^*)} - c_{\sigma(k^*)}| = 1 - \frac{k^*}{2n+1}. \tag{C.25}$$

Together with (C.23), this implies

$$c_{\sigma(k^*)} = 1 - b_{\sigma(k^*)}. \tag{C.26}$$

For any $i < k^*$, if $c_{\sigma(i)} = 1 - b_{\sigma(i)}$, then (C.23) implies that

$$f(x^*) \geq |x^*_{\sigma(i)} - c_{\sigma(i)}| = 1 - \frac{i}{2n+1} > 1 - \frac{k^*}{2n+1}, \tag{C.27}$$

which contradicts with the assumption that $f(x^*) = 1 - \frac{k^*}{2n+1}$. Therefore, we must have

$$c_{\sigma(i)} = b_{\sigma(i)} \quad \forall\, i \in \{1, \dots, k^* - 1\}. \tag{C.28}$$

Recall that the decreasing order of $\{x_1, \dots, x_n, 1 - x_1, \dots, 1 - x_n\}$ is

$$\begin{aligned} &b_{\sigma(1)}x_{\sigma(1)} + (1 - b_{\sigma(1)})(1 - x_{\sigma(1)}) \geq \cdots \geq b_{\sigma(n)}x_{\sigma(n)} + (1 - b_{\sigma(n)})(1 - x_{\sigma(n)}) \\ &\quad \geq (1 - b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1 - x_{\sigma(n)}) \geq \cdots \geq (1 - b_{\sigma(1)})x_{\sigma(1)} + b_{\sigma(1)}(1 - x_{\sigma(1)}). \end{aligned} \tag{C.29}$$

Based on (C.26), (C.28), and (C.29), we next prove

$$|x_{\sigma(k^*)} - c_{\sigma(k^*)}| \geq |x_{\sigma(i)} - c_{\sigma(i)}| \quad \forall\, i \in [n]. \tag{C.30}$$

If (C.30) holds, it implies

$$f(x) = \max_{i \in [n]} |x_i - c_i| = |x_{\sigma(k^*)} - c_{\sigma(k^*)}|. \tag{C.31}$$

If $b_{\sigma(k^*)} = 0$, then (C.26) implies $c_{\sigma(k^*)} = 1$, (C.31) implies $f(x) = 1 - x_{\sigma(k^*)}$, and the output in Line 4 satisfies

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) = 1 - x_{\sigma(k^*)} = f(x); \tag{C.32}$$

If $b_{\sigma(k^*)} = 1$, then (C.26) implies $c_{\sigma(k^*)} = 0$, (C.31) implies $f(x) = x_{\sigma(k^*)}$, and the output in Line 4 satisfies

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) = x_{\sigma(k^*)} = f(x). \tag{C.33}$$

The correctness of Line 4 follows.

It remains to prove (C.30). We divide its proof into two parts:

- Suppose $i < k^*$. By (C.29), we have

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) \geq (1 - b_{\sigma(i)})x_{\sigma(i)} + b_{\sigma(i)}(1 - x_{\sigma(i)}). \tag{C.34}$$

  - If $b_{\sigma(k^*)} = 0$ and $b_{\sigma(i)} = 0$, we have $c_{\sigma(k^*)} = 1$ and $c_{\sigma(i)} = 0$ by (C.26) and (C.28), respectively; (C.34) reduces to $1 - x_{\sigma(k^*)} \geq x_{\sigma(i)}$;
  - If $b_{\sigma(k^*)} = 0$ and $b_{\sigma(i)} = 1$, we have $c_{\sigma(k^*)} = 1$ and $c_{\sigma(i)} = 1$ by (C.26) and (C.28), respectively; (C.34) reduces to $1 - x_{\sigma(k^*)} \geq 1 - x_{\sigma(i)}$;
  - If $b_{\sigma(k^*)} = 1$ and $b_{\sigma(i)} = 0$, we have $c_{\sigma(k^*)} = 0$ and $c_{\sigma(i)} = 0$ by (C.26) and (C.28), respectively; (C.34) reduces to $x_{\sigma(k^*)} \geq x_{\sigma(i)}$;
  - If $b_{\sigma(k^*)} = 1$ and $b_{\sigma(i)} = 1$, we have $c_{\sigma(k^*)} = 0$ and $c_{\sigma(i)} = 1$ by (C.26) and (C.28), respectively; (C.34) reduces to $x_{\sigma(k^*)} \geq 1 - x_{\sigma(i)}$.

  In each case, the resulting expression is exactly (C.30). Overall, we see that (C.30) is always true when $i < k^*$.

- Suppose $i > k^*$. By (C.29), we have

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) \geq b_{\sigma(i)}x_{\sigma(i)} + (1 - b_{\sigma(i)})(1 - x_{\sigma(i)}); \tag{C.35}$$
$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) \geq (1 - b_{\sigma(i)})x_{\sigma(i)} + b_{\sigma(i)}(1 - x_{\sigma(i)}). \tag{C.36}$$

  - If $b_{\sigma(k^*)} = 0$, we have $c_{\sigma(k^*)} = 1$ by (C.26); (C.35) and (C.36) give $1 - x_{\sigma(k^*)} \geq \max\{x_{\sigma(i)}, 1 - x_{\sigma(i)}\}$;
  - If $b_{\sigma(k^*)} = 1$, we have $c_{\sigma(k^*)} = 0$ by (C.26); (C.35) and (C.36) give $x_{\sigma(k^*)} \geq \max\{x_{\sigma(i)}, 1 - x_{\sigma(i)}\}$.

  Both cases imply (C.30), so we see this also holds for $i > k^*$.

The same proof works when $k^* = n + 1$. In this case, there is no $i \in [n]$ such that $i > k^*$; on the other hand, when $i < k^*$, we replace (C.34) by

$$(1 - b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1 - x_{\sigma(n)}) \geq (1 - b_{\sigma(i)})x_{\sigma(i)} + b_{\sigma(i)}(1 - x_{\sigma(i)}), \tag{C.37}$$

and the argument proceeds unchanged. □

C.3 Optimality of Theorem 3.3

In this section, we prove that the lower bound in Theorem 3.3 is optimal (up to poly-logarithmic factors in n) for the max-norm optimization problem:

Theorem C.1. *Let $f_c\colon [0,1]^n \to [0,1]$ be an objective function for the max-norm optimization problem (Definition 3.2). Then there exists a quantum algorithm that outputs an $\tilde{x} \in [0,1]^n$ satisfying (3.16) with $\epsilon = 1/3$ using $O(\sqrt{n}\log n)$ quantum queries to O_f, with success probability at least 0.9.*

In other words, the quantum query complexity of the max-norm optimization problem is $\tilde{\Theta}(\sqrt{n})$.

We prove Theorem C.1 also using search with wildcards (Theorem 3.1).

Proof. It suffices to prove that one query to the wildcard query model O_c in (3.1) can be simulated by one query to O_{f_c}, where the c in (3.14) is the string c in the wildcard query model.

Assume that we query (T, y) using the wildcard query model. Then we query $O_{f_c}(x^{(T,y)})$ where for all $i \in [n]$,

$$x_i^{(T,y)} = \begin{cases} \frac{1}{2} & \text{if } i \notin T; \\ 0 & \text{if } i \in T \text{ and } y_i = 0; \\ 1 & \text{if } i \in T \text{ and } y_i = 1. \end{cases} \tag{C.38}$$

If $c_{|T} = y$, then

- if $|T| = n$ (i.e., $T = [n]$), then

$$f_c(x) = \max_{i \in [n]} |x_i^{(T,y)} - c_i| = 0 \tag{C.39}$$

  because for any $i \in [n]$, $x_i^{(T,y)} = y_i = c_i$;

- if $|T| \leq n - 1$, then

$$f_c(x) = \max_{i \in [n]} |x_i^{(T,y)} - c_i| + g_i = \frac{1}{2}, \tag{C.40}$$

  because for all $i \in T$ we have $x_i^{(T,y)} = y_i = c_i$ and hence $|x_i^{(T,y)} - c_i| = 0$, and for all $i \notin T$ we have $|x_i^{(T,y)} - c_i| = |\frac{1}{2} - c_i| = \frac{1}{2}$.

Therefore, if $c_{|T} = y$, then we must have $f_c(x^{(T,y)}) \in \{0, \frac{1}{2}\}$.

On the other hand, if $c_{|T} \neq y$, then there exists an $i_0 \in T$ such that $c_{i_0} \neq y_{i_0}$. This implies $x_{i_0}^{(T,y)} = 1 - c_{i_0}$; as a result, $f_c(x^{(T,y)}) = 1$ because on the one hand $f_c(x^{(T,y)}) \geq |1 - c_{i_0} - c_{i_0}| = 1$, and on the other hand $f_c(x^{(T,y)}) \leq 1$ as $|x_i^{(T,y)} - c_i| \leq 1$ for all $i \in [n]$.

Notice that the sets $\{0, \frac{1}{2}\}$ and $\{1\}$ do not intersect. Therefore, after we query $O_{f_c}(x^{(T,y)})$ and obtain the output, we can tell $Q_s(T, y) = 1$ in (3.1) if $O_{f_c}(x^{(T,y)}) \in \{0, \frac{1}{2}\}$, and output $Q_s(T, y) = 0$ if $O_{f_c}(x^{(T,y)}) = 1$. In all, this gives a simulation of one query to the wildcard query model O_c by one query to O_{f_c}.

As a result of Theorem 3.1, there is a quantum algorithm that outputs the c in (3.14) using $O(\sqrt{n} \log n)$ quantum queries to O_f. If we take $\tilde{x} = c$, then $f_c(\tilde{x}) = \max_i |c_i - c_i| = 0$, which is actually the optimal solution with $\epsilon = 0$ in (3.16). This establishes Theorem C.1. □