Quantum speedups for linear programming via interior point methods

Simon Apers, Sander Gribling

Quantum speedups for linear programming via interior point methods

Simon Apers* Sander Gribling†

Abstract

We describe a quantum algorithm based on an interior point method for solving a linear program with n inequality constraints on d variables. The algorithm explicitly returns a feasible solution that is ε-close to optimal, and runs in time $\sqrt{n}\,\mathrm{poly}(d, \log(n), \log(1/\varepsilon))$ which is sublinear for tall linear programs (i.e., $n \gg d$). Our algorithm speeds up the Newton step in the state-of-the-art interior point method of Lee and Sidford [FOCS '14]. This requires us to efficiently approximate the Hessian and gradient of the barrier function, and these are our main contributions.

To approximate the Hessian, we describe a quantum algorithm for the *spectral approximation* of A^TA for a tall matrix $A \in \mathbb{R}^{n\times d}$. The algorithm uses leverage score sampling in combination with Grover search, and returns a δ-approximation by making $O(\sqrt{nd}/\delta)$ row queries to A. This generalizes an earlier quantum speedup for graph sparsification by Apers and de Wolf [FOCS '20]. To approximate the gradient, we use a recent quantum algorithm for multivariate mean estimation by Cornelissen, Hamoudi and Jerbi [STOC '22]. While a naive implementation introduces a dependence on the condition number of the Hessian, we avoid this by pre-conditioning our random variable using our quantum algorithm for spectral approximation.

*Université Paris Cité, CNRS, IRIF, Paris, France. apers@irif.fr

†Tilburg University, Tilburg, the Netherlands. s.j.gribling@tilburguniversity.edu

Contents

1 Introduction

We consider the fundamental task of solving linear programs (LPs) of the form

$$\min_x c^T x \quad \text{s.t.} \quad Ax \geq b$$

where $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, $c \in \mathbb{R}^d$, and the minimization is over $x \in \mathbb{R}^d$. Famously, Khachiyan [27] showed how to use the ellipsoid method to solve such linear programs efficiently, i.e., in polynomial time. The result however was mostly of theoretical relevance, because the ellipsoid method runs very slow in practice. It was a second breakthrough result by Karmarkar [24] that introduced *interior point methods* (IPMs) as a provably polynomial time algorithm for solving LPs that is also efficient in practice. To this day the study of IPMs is central to our understanding of fast algorithms for solving LPs. Indeed, in the regime that we focus on, the classical state-of-the-art asymptotic complexity for solving an LP is achieved by an IPM of van den Brand, Lee, Sidford and Song [11], which runs in time $\widetilde{O}(nd + d^3)$.

With the growing interest in using quantum computers to solve optimization tasks, it is natural to ask whether quantum algorithms can speed up IPMs. In a nutshell, we give a positive answer for IPMs that solve "tall" LPs. We describe a quantum IPM that solves (full-dimensional) linear programs with d variables and n inequality constraints in time $\sqrt{n}\,\mathrm{poly}(d, \log(1/\varepsilon))$, given quantum query access to the problem data. This is sublinear in the input size when the LP is tall (n much larger than d). To obtain our quantum speedup we develop two novel quantum subroutines:

(i) an algorithm to obtain a spectral approximation of $B^T B$ for tall $B \in \mathbb{R}^{n \times d}$ (i.e. $n \gg d$), and

(ii) an approximate matrix-vector product algorithm.

While these new routines have their own independent interest (e.g., for computing statistical leverage scores), we use them to approximate the costly Newton steps at the core of an IPM. More specifically, we use these algorithms to approximate Hessians and gradients of three canonical *self-concordant barriers*. This enables us to perform an approximate Newton step, which is at the core of IPMs.

For comparison, and as one of the motivations for this work, we mention that a large number of works [25, 26, 7, 37, 20, 22] have recently studied the use of quantum linear system solvers combined with tomography methods to speed up the costly Newton step at the core of an IPM. These methods inherently introduce a dependence on the condition number of the linear system that they solve. A main feature of our work is that we do not introduce such a dependence. A second main feature of our work is that we achieve a $\log(1/\varepsilon)$-dependence, whereas previous quantum algorithms for LPs usually achieved a $\mathrm{poly}(1/\varepsilon)$-dependence [10, 3, 12, 2, 4, 8]. Recently, iterative refinement techniques have been used in the context of quantum IPMs to improve the error dependence to $\mathrm{polylog}(1/\varepsilon)$, but the dependence on condition numbers remains [37, 6].

We first give a brief overview of the type of IPMs we use and the landmark results in that area, focusing only on the theoretical developments and not the practical contributions. We then describe our quantum algorithms in more detail. Finally we describe our main results and put them in context.

1.1 Interior point methods for linear programs

In a seminal work, Nesterov and Nemirovskii linked interior point methods to *self-concordant barriers* [39]. We defer their definition until later – for this introduction one can think of them as convex functions that go to infinity as we approach the boundary of the feasible region; a standard example is the logarithmic barrier $f(x) = -\sum_{i=1}^{n} \log(a_i^T x - b_i)$ where a_i^T is the ith row of A. Given a barrier f, one can define the family of unconstrained problems parameterized by $\eta > 0$

$$\min_x f_\eta(x) \quad \text{where } f_\eta(x) = \eta c^T x + f(x).$$

For small η, the function effectively only takes into account the constraints (i.e., the barrier function) and the minimizer will be close to some geometric "center" of the feasible region, while for η sufficiently large the minimizer approaches that of the LP. Optimizing f_η for a fixed value of η is typically done using *Newton's method*. Newton's method approaches the optimum by making updates of the form $x' = x + n(x)$, with Newton step

$$n(x) = -H(x)^{-1} g(x), \tag{1}$$

where $H(x) = \nabla^2 f(x)$ is the Hessian at x, and $g(x) = \nabla f(x)$ is the gradient at x. Self-concordant functions f have the following desirable property: if we are close to a minimizer for f_η, then we can multiplicatively increase η by a small amount to η' and still quickly converge to a minimizer of the new $f_{\eta'}$ using (approximate) Newton steps. The complexity of such an IPM is thus determined by the rate at which we can increase η and the cost of an (approximate) Newton step.

A self-concordant barrier f has an associated complexity parameter ϑ_f that governs this rate: we can increase η multiplicatively with a factor roughly $1 + 1/\sqrt{\vartheta_f}$. Starting from a point close to the minimizer of f_1, the number of times we need to increase η scales roughly as $\widetilde{O}(\sqrt{\vartheta_f}\log(1/\varepsilon))$.[1]

The standard *logarithmic barrier* used in [41] has a complexity of n, the number of inequality constraints. In this work, we focus on the regime where $n \gg d$. In that setting, Vaidya showed how to obtain an improved complexity using the *volumetric barrier* [47] – a linear combination of that barrier with the logarithmic barrier has an improved complexity of $O(\sqrt{nd})$ [48]. In their seminal work, Nesterov and Nemirovskii showed that each closed convex domain in $\mathbb{R}^d$ admits a *universal barrier* whose complexity is $O(d)$ [39]. The iterations of an IPM based on the universal barrier are unfortunately not efficiently computable. Roughly 20 years later, Lee and Sidford defined a self-concordant barrier based on Lewis weights, the *Lewis weight barrier*, which has complexity $O(d\,\mathrm{polylog}(n))$ [31]; based on this barrier they designed an IPM that uses $O(\sqrt{d}\log(1/\varepsilon)$ iterations, each of which involves solving $\widetilde{O}(1)$ linear systems. We summarize the three landmark barriers and their Hessians and gradients in Table 1. We define and discuss leverage scores and Lewis weights in more detail in Section 1.2.2.

In Section 6 we argue that it suffices to implement approximations of the Newton steps, based on approximations of the Hessians and gradients in Table 1. Specifically, we will implement approximate Newton steps of the form $\tilde{n}(x) = -Q(x)^{-1}\tilde{g}(x)$, where

$$Q(x) \approx H(x) \quad \text{and} \quad \|\tilde{g}(x) - g(x)\|_{H(x)^{-1}} \leq \delta, \tag{2}$$

[1] Throughout we use $\widetilde{O}$-notation to hide polylogarithmic factors in n, d, and $1/\varepsilon$. We assume each entry of A, b, c has bit-complexity $\mathrm{polylog}(n, d)$.

Barrier	Complexity ϑ	Hessian $H(x)$	Gradient $g(x)$
Logarithmic barrier	n	$A^T S_x^{-2} A$	$-A^T S_x^{-1}\mathbf{1}$
Volumetric / hybrid barrier	$\sqrt{nd}$	$\approx A^T S_x^{-1}\Sigma_x S_x^{-1} A$	$-A^T S_x^{-1}\Sigma_x\mathbf{1}$
ℓ_p-Lewis weight barrier	$d \cdot \mathrm{polylog}(n)$	$\approx A^T S_x^{-1} W_x S_x^{-1} A$	$-A^T S_x^{-1} W_x\mathbf{1}$

Table 1: Self-concordant barriers, their complexity as a function of the number of variables d and constraints n, and expressions for the Hessian and gradient. Diagonal matrices S_x, Σ_x, and W_x contain in the ith diagonal entry respectively the ith slack $(Ax-b)_i$, the leverage score $\sigma_i(S_x^{-1}A)$, or the ℓ_p-Lewis weight $w_i^{(p)}(S_x^{-1}A)$. $A \approx B$ is shorthand for $A \preceq B \preceq \widetilde{O}(1)A$.

for $1/\delta \in \widetilde{O}(1)$, where we used the shorthand $\|v\|_A$ to denote $\sqrt{v^T A v}$ and $A \approx B$ for $A \preceq B \preceq \widetilde{O}(1)A$.

1.2 Quantum algorithms

Here we summarize the different quantum algorithms that we combine to approximate Newton steps in the sense of Eq. (2).

1.2.1 Quantum spectral approximation

As mentioned before, our first task is to spectrally approximate a matrix $B^T B$ where $B \in \mathbb{R}^{n\times d}$ and $n \gg d$. We prove the following theorem, where a row query to B corresponds to learning all the entries of a row of B.

Theorem 3.1 (Quantum spectral approximation)**.** *Consider query access to a matrix $B \in \mathbb{R}^{n\times d}$ with row sparsity r. For any $0 < \varepsilon \leq 1$, there is a quantum algorithm that, with high probability, returns a matrix $\widetilde{B} \in \mathbb{R}^{\widetilde{O}(d/\varepsilon^2)\times d}$ satisfying*

$$(1-\varepsilon)\widetilde{B}^T\widetilde{B} \preceq B^T B \preceq (1+\varepsilon)\widetilde{B}^T\widetilde{B},$$

while making $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to B, and taking time $\widetilde{O}(r\sqrt{nd}/\varepsilon + d^\omega)$.

For comparison, the classical state of the art is $\widetilde{O}(\mathrm{nnz}(B) + d^3)$ where $\mathrm{nnz}(B)$ is the number of non-zero entries of B [15]. We thus achieve a quantum speedup for dense instances that have $n \gg d$. It is not hard to see that any classical algorithm for spectral approximation must make $\Omega(n)$ row queries to B.[2] In a nutshell, our $n \to \sqrt{nd}$ quantum speedup comes from using Grover search to find the $\widetilde{O}(d)$ important rows among the n rows of B in $\widetilde{O}(\sqrt{nd})$ row queries to B.

In this work, we will use this in the context of IPMs to approximate the Hessian at each iterate. As can be seen in Table 1, the Hessians that we consider take the form $H = B^T B$, with $B \in \mathbb{R}^{n\times d}$ equal to some diagonal matrix of weights times the constraint matrix A.

More broadly, such spectral approximations are central to dimensionality reduction techniques in several areas. For example, in data science, where B represents a tall data matrix, a spectral approximation of $B^T B$ can be used for statistical regression [30, 40]. In graph theory, spectral

[2] E.g., consider $B \in \mathbb{R}^{n\times 1}$ with B either a vector with all zeros, or a vector with all zeros but a single 1.

approximations can be used to sparsify a graph. We note that the above theorem generalizes an earlier quantum algorithm for graph sparsification [5], although it uses very different techniques. Indeed, we can recover the complexity in [5, Theorem 1] by letting $A \in \mathbb{R}^{n \times d}$ denote the edge-vertex incidence matrix of a graph with n edges and d vertices, which has sparsity $r = 2$. See Remark 3.3 for more details.

At a high level, our algorithm does row sampling based on statistical *leverage scores*. The leverage score of the i-th row b_i^T of B is defined as

$$\sigma_i(B) = b_i^T (B^T B)^+ b_i,$$

and this measures the "statistical importance" of a row with respect to the full set of rows [36, 34]. Here C^+ denotes the Moore-Penrose pseudoinverse of C. A matrix $\widetilde{B}$ is then constructed by sampling and rescaling a subset of $\widetilde{O}(d)$ rows of B, in such a way that ensures $\widetilde{B}^T \widetilde{B} \approx B^T B$. Typically one includes each row i independently with a probability that is roughly proportional to $\sigma_i(B)$. Naively computing these scores $\sigma_i(B)$ however already requires knowledge of $B^T B$. We bypass this issue by building on a recent, elegant bootstrapping technique by Cohen, Lee, Musco, Musco, Peng and Sidford [16]. Implementing their technique in the quantum setting, in sublinear time, is challenging and one of our main contributions.

As an example application, we can use this theorem to get a spectral approximation of the Hessian $H(x) = A^T S_x^{-2} A$ of the logarithmic barrier. To this end, we apply Theorem 3.1 to the matrix $S_x^{-1} A$, which gives us an approximation $\widetilde{H} \approx H(x)$ while making only $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to $S_x^{-1} A$. Recalling that S_x is a diagonal matrix with entries $(S_x)_{ii} = (Ax - b)_i$, it is clear that we can simulate a row query to $S_x^{-1} A$ with a single query to b and a single row query to A.

For the volumetric barrier and the Lewis weight barrier, we need query access to the rescaled matrices $\Sigma_x^{1/2} S_x^{-1} A$ and $W_x^{1/2} S_x^{-1} A$, respectively. The rescalings by Σ_x and W_x are based on leverage scores and Lewis weights, respectively, and we have to describe additional quantum algorithms for approximating these. Our algorithm to approximate leverage scores is a direct consequence of Theorem 3.1: if we have a spectral approximation $\widetilde{B}$ such that $\widetilde{B}^T \widetilde{B} \approx_\varepsilon B^T B$, we also get that

$$b_i^T (\widetilde{B}^T \widetilde{B})^+ b_i \approx_\varepsilon b_i^T (B^T B)^+ b_i = \sigma_i(B).$$

Hence, we can use a spectral approximation of B to estimate its leverage scores. Lewis weights are more complicated to define and approximate, we describe how to do so in the next section. Based on these algorithms we obtain the complexities for Hessian approximation in Table 2.

Barrier	Row queries	Time
Logarithmic barrier	$\sqrt{nd}$	$r\sqrt{nd} + d^\omega$
Volumetric barrier	$\sqrt{nd}$	$r\sqrt{nd} + d^\omega$
Lewis weight barrier	$\sqrt{n}d^{3/2}$	$d^{3/2}(r^2\sqrt{nd} + d^\omega)$

Table 2: Complexity of quantum algorithms for $\widetilde{O}(1)$-spectral approximations of Hessians. Row queries are to $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, r is the row-sparsity of A, ω is the matrix multiplication coefficient, and we ignore polylog-factors.

1.2.2 Lewis weights

To approximate Lewis weights, we first note that leverage scores and spectral approximation are implicitly defined with respect to the ℓ_2-norm: $\widetilde{B}$ is a spectral approximation of B if and only if $\|\widetilde{B}x\|_2 \approx \|Bx\|_2$ for all vectors x. If we extend this notion from the ℓ_2-norm to the ℓ_p-norm, we naturally arrive at the notion of ℓ_p-Lewis weights $\{w_i^{(p)}(B)\}$ of a matrix B [33]. As is shown in [17], these can be used to obtain an ℓ_p-approximation $\widetilde{B}$ of B in the sense that $\|\widetilde{B}x\|_p \approx \|Bx\|_p$ for all x. The ℓ_p-Lewis weights are implicitly defined through a fixed-point equation, they are defined as the leverage scores of a rescaling of B based on the Lewis weights:

$$w_i^{(p)}(B) = \sigma_i(W^{1/2-1/p}B), \tag{3}$$

where W is the diagonal matrix with entries $(W)_{ii} = w_i^{(p)}(B)$. While more challenging then leverage scores, classical algorithms for approximating the Lewis weights have been described in [30, 17, 21]. These algorithms typically obtain the Lewis weights through an iterative process of computing regular leverage scores, and rescaling rows based on these scores. We follow such a procedure from [30]. However, since we are aiming for a sublinear runtime scaling as $\sqrt{n}\,\mathrm{poly}(d)$, we have to be more careful since we cannot afford to even write down all Lewis weights. Instead, we construct implicit data structures that allow us to (relatively) efficiently query the intermediate weights. We obtain the following theorem, which we believe is of independent interest.[3]

Theorem (Quantum Lewis weights, informal version of Theorem 4.5)**.** *Consider query access to a matrix $B \in \mathbb{R}^{n\times d}$ with row sparsity r. For any $0 < \varepsilon \leq 1$ and $p \geq 2$, there is a quantum algorithm that provides query access to ε-multiplicative approximations of the Lewis weights $w_i^{(p)}(B)$. The algorithm has a preprocessing phase that makes $\widetilde{O}(\sqrt{n}d^{3/2}/\varepsilon^2)$ row queries to A and takes time $\sqrt{n}\cdot\mathrm{poly}(d,\log(n),1/\varepsilon)$. After this, each query requires 1 row query to B and time $\widetilde{O}(r^2d^{1/2}/\varepsilon)$.*

1.2.3 Quantum gradient and matrix-vector approximation

We now turn to the approximation of the gradient, which we need to approximate in the "inverse-local norm", i.e., find $\tilde{g}$ such that $\|\tilde{g}-g\|_{H^{-1}} \leq \delta$. We first note that, if $B^TB \preceq H$ then $H^{-1} \preceq (B^TB)^{-1}$, and therefore it suffices to find approximations of the gradient in the $(B^TB)^{-1}$-norm where B is as in the previous section. An important observation is that, for these B's, each of the gradients in Table 1 take the form $g = B^Tv$ where v is either $\mathbf{1}$, $\Sigma_x^{1/2}\mathbf{1}$, or $W_x^{1/2}\mathbf{1}$. Our goal is thus, given query access to B and v, to find a vector $\tilde{y}$ such that

$$\|\tilde{y} - B^Tv\|_{(B^TB)^{-1}} \leq \delta.$$

To do so, we use a recent quantum algorithm for multivariate mean estimation by Cornelissen, Hamoudi and Jerbi [18]. For a random variable Y with mean μ and covariance matrix Σ, this returns an estimate $\tilde{\mu}$ satisfying $\|\tilde{\mu}-\mu\|_2 \leq \delta$ while using only $\widetilde{O}(\sqrt{d\,\mathrm{Tr}(\Sigma)}/\delta)$ samples of Y (as compared to $O(\mathrm{Tr}(\Sigma)/\delta^2)$ samples that are required classically).[4]

It is natural to rewrite $g = B^Tv$ as the mean μ of a random variable $Y = nv_iB^Te_i$, with $i \in [n]$ uniformly at random, and then apply quantum mean estimation to Y. However, it is not clear how to

[3] For the important special case of leverage scores ($p = 2$) we obtain significantly better bounds on the complexity, see Lemmas 3.7 and 3.9.

[4] The quantum bound stated here assumes that $\mathrm{Tr}(\Sigma)/\delta^2 \geq d$, which will be satisfied in our applications.

control the trace of the corresponding covariance matrix Σ, one can only show $\Sigma \preceq nB^T \operatorname{Diag}(v)^2 B$, and naively turning the guarantee on $\|\tilde{\mu} - \mu\|_2$ into the required guarantee on $\|\tilde{\mu} - \mu\|_{H^{-1}}$ would introduce an unwanted dependence on the condition number of H.

We avoid this problem by first constructing a spectral approximation $\widetilde{H} \approx_\delta H$ using Theorem 3.1. This allows us to sample from the "pre-conditioned" random variable $Y = nv_i \widetilde{H}^{-1/2} B^T e_i$ with $i \in [n]$ uniformly at random. Now note that $\Sigma = \mathbb{E}[YY^T] = n\widetilde{H}^{-1/2} B^T B \widetilde{H}^{-1/2} \approx nI_d$, and moreover

$$\delta \geq \|\tilde{\mu} - \mu\|_2 = \|\widetilde{H}^{1/2}\tilde{\mu} - B^T v\|_{\widetilde{H}^{-1}} \approx \|\widetilde{H}^{1/2}\tilde{\mu} - B^T v\|_{H^{-1}},$$

so that returning the estimate $\widetilde{H}^{1/2}\tilde{\mu}$ yields a valid estimator of the gradient. This trick avoids any condition number dependence, and it leads to the following theorem.

Theorem 5.1 (Approximate matrix-vector product)**.** *Assume query access to a vector $v \in \mathbb{R}^n$ and a matrix $B \in \mathbb{R}^{n \times d}$ with row sparsity r. There is a quantum algorithm that returns a vector $\tilde{y}$ satisfying*

$$\|\tilde{y} - B^T v\|_{(B^T B)^{-1}} \leq \delta,$$

while making $\widetilde{O}(\sqrt{n}d\|v\|_\infty/\delta)$ row queries to B and v, and taking time $\widetilde{O}(r\sqrt{n}d^2\|v\|_\infty/\delta + d^\omega)$.

Applying this theorem to the various barrier functions, we obtain the complexities in Table 3 below.

Barrier	Row queries	Time
Logarithmic barrier	$\sqrt{n}d$	$r\sqrt{n}d^2 + d^\omega$
Volumetric barrier	$\sqrt{n}d$	$r\sqrt{n}d^2 + d^\omega + d \cdot \min\{d^\omega, dr^2\}$
Lewis weight barrier	$\sqrt{n}d^{5/2}$	$r^2\sqrt{n}d^{7/2} + d^{\omega+3}$

Table 3: Complexity of quantum algorithms for estimating gradients up to constant error. Row queries are to $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, r is the row-sparsity of A, and we ignore polylog-factors.

1.3 Main result and discussion

We summarize our main result, which is based on the Lewis weight IPM. Combining our quantum algorithms for spectral approximation, Lewis weight computation, and approximate matrix-vector multiplication, we obtain the following theorem.

Theorem 1.1 (Quantum IPM for LP)**.** *Consider an LP*[5] $\min c^T x$ *s.t.* $Ax \geq b$*. Assume $A \in \mathbb{R}^{n \times d}$ has row sparsity r, and let* val *denote its optimal value. Given query access to A, b, c, we give a quantum interior point method that explicitly returns a vector $\tilde{x}$ satisfying*

$$A\tilde{x} \geq b \quad \text{and} \quad c^T\tilde{x} \leq \text{val} + \varepsilon.$$

Depending on the barrier function, the algorithm makes $\widetilde{O}(nd)$ (logarithmic), $\widetilde{O}(n^{3/4}d^{5/4})$ (volumetric) or $\widetilde{O}(\sqrt{n}d^3)$ (Lewis weight) row queries to A, b and c. For the Lewis weight barrier, the time complexity is $\widetilde{O}(\sqrt{n}\operatorname{poly}(d))$.

[5]We assume that the LP is full-dimensional, and that we are given an initial point x_0 in the interior of the feasible region, not too close to the boundary. See Section 6 for a more precise statement.

The quantum IPM based on the Lewis weight barrier uses a sublinear number of row queries to A when n is much larger than d (specifically, $n \geq \gamma d^6$ for some constant γ). Examples of such overconstrained LPs include Chebyshev approximation and linear separability (see e.g. [14]). In the following we put our results in perspective by (i) describing a lower bound, and (ii) describing a benchmark algorithm based on cutting plane methods.

The lower bound essentially follows from [3], and we present it in Section 8. The bound states that any quantum algorithm for solving an LP to constant precision must make $\Omega(\sqrt{nd})$ row queries. This shows that the $\sqrt{n}$-dependency in our algorithm is optimal, while it leaves much room for improvement regarding the d-dependency. Indeed, *we conjecture that this lower bound is tight*, and we later give some hopeful directions to improve our results towards such a bound.

We now compare our results to a quantum speedup over *cutting plane methods*. In Appendix A we show that such methods are particularly amenable to a Grover-type quantum speedup. Indeed, they are usually phrased in terms of a "separation oracle" that returns an arbitrary violated constraint, and this can be implemented quadratically faster using Grover search. The state-of-the-art (and highly technical) cutting plane method from Lee, Sidford and Wong [32] makes $\widetilde{O}(d)$ queries to a separation oracle, and this directly yields a quantum algorithm that makes $\widetilde{O}(\sqrt{n}d)$ row queries to A (and requires additional time $\widetilde{O}(r\sqrt{n}d + d^3)$). The query complexity is a factor $\sqrt{d}$ above the lower bound, and such a black-box approach building on a classical algorithm with separation queries cannot further improve over this (indeed, it is not hard to see that any classical algorithm for LP solving must make $\Omega(d)$ separation queries). We see this as a clear motivation for looking at "white-box" quantum algorithms, such as ours based on IPMs. Indeed, also classically the state-of-the-art for solving tall, dense, LPs is achieved by an IPM [11]; it runs in time $\widetilde{O}(nd + d^3)$.

Finally, we present some of the open directions to improve over our work. The first two relate to the complexity of approximating a single Newton step.

1. **Cheaper gradient approximation:** We can spectrally approximate the Hessian with $\widetilde{O}(\sqrt{nd})$ row queries for the logarithmic and the volumetric barrier. This is tight, and it matches the lower bound for LP solving. However, our algorithm for approximating the gradient of the simplest, logarithmic barrier already has complexity $\widetilde{O}(\sqrt{n}d)$. This is a clear direction to improve the complexity of our quantum IPM. To surpass our bound, one could try to better exploit the structure of the gradients that we need to estimate, or change the path-following method so that a weaker approximation of the gradient would suffice.

2. **Low-precision leverage scores / Lewis weights:** Currently, the (time) complexity of approximate Newton steps for the volumetric and Lewis weight barriers is higher than that of the logarithmic barrier. The main reason for this is that, in order to obtain a good approximation of the gradient, we currently require high-precision $O(1/\sqrt{d})$-approximations of leverage scores or Lewis weights. Can one base an IPM with these barriers solely on $O(1)$-approximate leverage scores or Lewis weights?

   For Lewis weights such high-precision estimates are particularly costly. We use a stability result from [21] that roughly says that if (i) $w \approx_\varepsilon \sigma(W^{\frac{1}{2}-\frac{1}{p}}A)$ (i.e., it approximately satisfies Eq. (3)), then (ii) $w \approx_{\varepsilon\sqrt{d}} w^{(p)}(A)$. We use a $1/\varepsilon^3$-algorithm to obtain (i), and therefore we incur an additional factor $d^{3/2}$ to obtain a constant error in (ii). Can we improve the

poly$(1/\varepsilon)$-dependence of our algorithm? Or better yet, can we base an IPM on the first (fixed-point) notion of approximation? If so, this would be a significant step towards matching the $\Omega(\sqrt{nd})$ row query lower bound, at least in the cost of one iteration.

While the cost of a single Newton step could conceivably be improved to $\widetilde{O}(\sqrt{nd})$ row queries, an IPM still requires a large number of such steps (e.g., $\widetilde{O}(\sqrt{d})$ Newton steps for the Lewis weight barrier). Dealing with this overhead is another natural direction.

3. **Dynamic data structures:** Classical state-of-the-art algorithms often use dynamic data structures to *amortize* the cost of an expensive routine (e.g., computing a Newton step) over the number of iterations. Maintaining such a data structure in a sublinear time quantum algorithm is challenging and has not been explored much. One nice example is [8] where the authors introduce dynamic Gibbs sampling to improve the state-of-the-art first-order method for zero-sum games (and thereby also for LPs). IPMs also seem a natural setting in which to explore this question. For example, the Hessian changes slowly between iterations of an IPM by design. Whereas we show that the complexity of our spectral approximation algorithm is tight when applied to a single B^TB, it seems likely that this can be improved when considering a a sequence of slowly changing matrices.

Finally, there are some unexplored applications of our quantum algorithms.

4. **Further applications:** As a subroutine, we derived quantum algorithms for estimating leverage scores and Lewis weights. These quantities have their independent merit, e.g. for doing dimensionality reduction in data science. We leave it as future work to explore such applications, and to compare our Grover-based approach to e.g. the approaches in [43, 35] based on quantum linear system solving.

2 Preliminaries

Notation. We let $\mathcal{S}^d \subseteq \mathbb{R}^{d\times d}$ be the set of d-by-d symmetric matrices. For $H \in \mathcal{S}^d$ we write $H \succ 0$ (resp. $H \succeq 0$) if H is positive definite (resp. positive semidefinite). For vectors $u, v \in \mathbb{R}^d$ we let $\langle u, v\rangle = \sum_{i\in[d]} u_i v_i$ denote the standard inner product. A positive definite matrix H defines an inner product $\langle u, v\rangle_H \coloneqq \langle u, Hv\rangle$. We write $\|u\|_H = \sqrt{\langle u, u\rangle_H}$. We often use the *local norm* associated to a twice continuously differentiable function f whose Hessian $H(x)$ is positive definite for all x in its domain D. Given a fixed reference inner product $\langle\cdot,\cdot\rangle$ (e.g. the standard inner product on $\mathbb{R}^d$), f defines a family of *local* inner products, one for each x in its domain via its Hessian:

$$\langle u, v\rangle_{H(x)} \coloneqq \langle u, H(x)v\rangle.$$

We will write $\langle\cdot,\cdot\rangle_x$ instead of $\langle\cdot,\cdot\rangle_{H(x)}$ when the function f is clear from context.

For $\varepsilon > 0$, the matrix $Q \in \mathcal{S}^d$ is an ε-spectral approximation of a d-by-d PSD matrix $H \succeq 0$ if

$$(1-\varepsilon)Q \preceq H \preceq (1+\varepsilon)Q,$$

where $A \preceq B$ is equivalent to $B - A \succeq 0$. We denote such a spectral approximation by $Q \approx_\varepsilon H$. We will also use this notation for scalars, where $a \approx_\varepsilon b$ denotes $(1-\varepsilon)b \le a \le (1+\varepsilon)b$, and for vectors (where the inequalities hold entrywise).

Finally, for a vector $v \in \mathbb{R}^n$, we use the notation $V = \mathrm{Diag}(v)$ for the diagonal n-by-n matrix who's i-th diagonal element corresponds to the i-th element of v.

With high probability. Throughout this work, "with high probability" means with probability at least $1 - 1/n^c$ for an arbitrarily high but fixed constant $c > 0$, where n typically denotes the size of the problem instance.

Quantum computational model. We assume the usual quantum computational model (see e.g. [5]), which is a classical system that can (i) run quantum subroutines on $O(\log n)$ qubits, (ii) can make quantum queries to the input, and (iii) has access to a quantum-read/classical-write RAM (QRAM) of $\mathrm{poly}(n)$ bits.[6]

The *time complexity* of an algorithm in this model measures the number of elementary classical and quantum gates or QRAM operations that the algorithm uses. The *query complexity* of an algorithm measures the number of queries to the input. Note that all statements about query complexity are independent of the QRAM assumption (i.e., we can drop the QRAM assumption at the cost of a polynomial increase in the *time* complexity of the algorithm).

Regarding queries to the input, we will typically consider inputs consisting of tall matrices $Z \in \mathbb{R}^{n \times d}$ (where d can be 1). We count the query complexity in terms of the number of *row queries*, where a row query to the i-th row of Z returns the entire row. We note that a query can be made in superposition. The row query complexity can alternatively be expressed in the *sparse access* model (see e.g. [13, Section 2.4]) by noting that a single row query corresponds to r sparse queries, with r the row-sparsity of Z. We will frequently work with matrices of the form DZ for a diagonal matrix D, and we note that one row query to DZ can be reduced to 1 query to D and 1 row query to Z.

3 Quantum algorithm for spectral approximation

In this section we describe our main quantum algorithm for constructing a spectral approximation of a matrix $A \in \mathbb{R}^{n \times d}$ (where now A denotes a general matrix, not necessarily the LP constraint matrix). The algorithm has sublinear time and query complexity for sufficiently tall matrices. While motivated by interior point methods, it is a self-contained section, and we envision applications of these quantum algorithms in other areas such as statistical regression. The algorithm's complexity is described in the following theorem.

Theorem 3.1 (Quantum spectral approximation)**.** *Consider query access to a matrix $A \in \mathbb{R}^{n \times d}$ with row sparsity r. For any $0 < \varepsilon \leq 1$, there is a quantum algorithm that, with high probability, returns a matrix $B \in \mathbb{R}^{\widetilde{O}(d/\varepsilon^2) \times d}$ satisfying*

$$(1 - \varepsilon) B^T B \preceq A^T A \preceq (1 + \varepsilon) B^T B,$$

while making $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to A, and taking time $\widetilde{O}(r\sqrt{nd}/\varepsilon + d^\omega)$.

Based on this theorem, and with some additional work, we can derive the following theorem on quantum algorithms for approximating leverage scores. We will use this later on.

[6] We charge unit cost for classically writing a bit, or quantumly reading a bit (potentially in superposition).

Theorem 3.2 (Quantum leverage score approximation)**.** *Assume query access to a matrix $A \in \mathbb{R}^{n\times d}$ with row sparsity r. For any $0 < \varepsilon \leq 1$, there is a quantum algorithm that, with high probability, provides query access to estimates $\tilde{\sigma}_i$ for any $i \in [n]$ satisfying*

$$\tilde{\sigma}_i = (1 \pm \varepsilon)\sigma(A).$$

The cost of the algorithm is either of the following:

1. *Preprocessing: $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to A and $\widetilde{O}(r\sqrt{nd}/\varepsilon + d^\omega + \min\{d^\omega, dr^2/\varepsilon^2\}/\varepsilon^2)$ time. Cost per estimate $\tilde{\sigma}_i$: 1 row query to A and $O(r^2)$ time.*
2. *Preprocessing: $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to A and $\widetilde{O}(r\sqrt{nd}/\varepsilon + d^\omega + \min\{d^\omega, dr^2/\varepsilon^2\}/\varepsilon^2 + d^2/\varepsilon^4)$ time. Cost per estimate $\tilde{\sigma}_i$: 1 row query to A and $\widetilde{O}(r/\varepsilon^2)$ time.*

Proofs of both theorems are given in Section 3.2.4.

Remark 3.3 (Graph sparsification)**.** *We note that Theorem 3.1 generalizes an earlier quantum algorithm for graph sparsification [5], although it uses very different techniques. Indeed, we can recover the complexity in [5, Theorem 1] by letting $A \in \mathbb{R}^{n\times d}$ denote the edge-vertex incidence matrix of a graph with n edges and d vertices, which has sparsity $r = 2$. This directly recovers the $\widetilde{O}(\sqrt{nd}/\varepsilon)$ query complexity of [5]. To also recover the $\widetilde{O}(\sqrt{nd}/\varepsilon)$ time complexity of [5], note that the d^ω term in Theorem 3.1 is due to solving linear systems for matrices of the form $B^TB \in \mathbb{R}^{d\times d}$, where B contains a (rescaled) subset of $\widetilde{O}(d)$ rows of A. When A is an edge-vertex incidence matrix, then B^TB will be a Laplacian matrix with $\widetilde{O}(d)$ nonzero entries, and these can be solved in near-linear time $\widetilde{O}(d)$ using fast Laplacian solvers [45].*

3.1 Repeated halving algorithm

We base our quantum algorithm for spectral approximation on a simple, elegant, recursive sparsification algorithm described by Cohen, Lee, Musco, Musco, Peng and Sidford in [16]. As most sparsification algorithms, it requires subsampling rows. The simplest way of subsampling is just keeping any individual row with a probability p. We denote by

$$B \subseteq_p A$$

the matrix obtained by keeping every individual row of A independently with probability p. We will also use a more refined notion of subsampling. For a matrix $A \in \mathbb{R}^{n\times d}$ and an entrywise positive weight vector $w = (w_i)_{i\in[n]}$, we denote by

$$B \overset{w,\varepsilon}{\leftarrow} A$$

the matrix obtained by the following weighted subsampling process (where c is a large enough universal constant):

Algorithm 1: Weighted subsampling $B \overset{w,\varepsilon}{\leftarrow} A$

Input: $A \in \mathbb{R}^{n\times d}$, positive $w \in (\mathbb{R}_{>0} \cup \infty)^n$ and $\varepsilon > 0$

for $i \in [n]$ **do**
  with probability $p_i := \min\{1, cw_i \log(d)/\varepsilon^2\}$, add row $\frac{1}{\sqrt{p_i}} a_i^T$ to B;
end for
return B

The rescaling ensures that, irrespective of w, we have that if $B \overset{w,\varepsilon}{\leftarrow} A$ then

$$\mathbb{E}[B^T B] = \sum_i p_i \frac{1}{p_i} a_i a_i^T = \sum_i a_i a_i^T = A^T A.$$

We can ensure concentration of $B^T B$ around its expectation by picking the weights appropriately. A typical choice is based on *leverage scores*. The leverage score of the i-th row a_i^T of a matrix $A \in \mathbb{R}^{n \times d}$ is defined as

$$\sigma_i(A) := a_i^T (A^T A)^+ a_i.$$

The following lemma demonstrates that sampling probabilities based on leverage scores ensure concentration.

Lemma 3.4 (Consequence of the matrix Chernoff bound [46], [16, Lemma 4])**.** *Consider matrix $A \in \mathbb{R}^{n \times d}$ and weight vector $w = (w_i)_{i \in [n]}$ satisfying $w_i \geq \sigma_i(A)$. Then, with high probability, $B \overset{w,\varepsilon}{\leftarrow} A$ satisfies $B^T B \approx_\varepsilon A^T A$ and has at most $O(\sum_{i \in [n]} p_i)$ rows, where we recall that $p_i = \min\{1, c w_i \log(d)/\varepsilon^2\}$.*

Now note that we always have $\sum_{i \in [n]} p_i \leq c \log(d) \|w\|_1 / \varepsilon^2$ and

$$\sum_i \sigma_i(A) = \sum_i a_i^T (A^T A)^+ a_i = \mathrm{Tr}(A^T A (A^T A)^+) \leq d.$$

Hence, if we have good estimates $w_i \in \Theta(\sigma_i(A))$ of the leverage scores of A, then $\|w\|_1 \in O(d)$ and the resulting matrix B has only $O(d \log(d)/\varepsilon^2)$ rows. Of course, in general it is not clear how to obtain such good estimates of the leverage scores. We will follow the approach of [16], which introduces a slight generalization of leverage scores: the *generalized* leverage score of the i-th row of a matrix $A \in \mathbb{R}^{n \times d}$ with respect to a matrix $B \in \mathbb{R}^{n_B \times d}$ is defined by

$$\sigma_i^B(A) := \begin{cases} a_i^T (B^T B)^+ a_i & \text{if } a_i \perp \ker(B) \\ \infty & \text{otherwise.} \end{cases}$$

Note that (i) $\sigma_i(A) = \sigma_i^A(A)$ and (ii) if $B \subseteq_p A$ for $p \in [0,1]$ then $\sigma_i^B(A) \geq \sigma_i(A)$. The second statement shows that we can derive overestimates of the leverage scores from a uniformly subsampled matrix B of A. The following theorem from [16] shows that if $p = 1/2$ then these overestimates do not increase the resulting number of sampled rows significantly.

Theorem 3.5 ([16, Theorem 4])**.** *Consider $A \in \mathbb{R}^{n \times d}$ and $B \subseteq_{1/2} A$. Define $w \in \mathbb{R}^n$ by $w_i = \sigma_i^B(A)$. Then, with high probability, $\sum_{i \in [n]} p_i \in O(d \log(d)/\varepsilon^2)$, where we recall that $p_i = \min\{1, c w_i \log(d)/\varepsilon^2\}$. Consequently, if $\widetilde{B} \overset{w,\varepsilon}{\leftarrow} A$, then with high probability $\widetilde{B}^T \widetilde{B} \approx_\varepsilon A^T A$ and $\widetilde{B}$ has $O(d \log(d)/\varepsilon^2)$ rows.*

Of course, the uniform subsampling $A' \subseteq_{1/2} A$ only reduces the matrix size by a factor roughly two. In the following algorithm (based on [16, Algorithm 2]), a chain of $L \in O(\log(n/d))$ uniformly downsampled matrices $A_1, \ldots, A_L$ is considered, which does result in a matrix A_L with only $\widetilde{O}(d)$ rows. By Theorem 3.5 we can then derive leverage scores from (an approximation B_ℓ of) A_ℓ to construct an approximation $B_{\ell-1}$ of $A_{\ell-1}$. Repeating this process L times finally yields an approximation B_0 of $A_0 = A$.

Algorithm 2: Repeated halving algorithm

Input: matrix $A \in \mathbb{R}^{n \times d}$, approximation factor $\varepsilon > 0$

1 let $A_L \subseteq_{1/2} \cdots \subseteq_{1/2} A_1 \subseteq_{1/2} A$ for $L = \lceil \log_2(n/d) \rceil$; let $B_L = A_L$;
2 **for** $\ell = L-1, L-2, \ldots, 1$ **do**
3 | let $B_\ell \overset{w,1/2}{\leftarrow} A_\ell$ with $2\sigma_i^{B_{\ell+1}}(A_\ell) \le w_i \le 4\sigma_i^{B_{\ell+1}}(A_\ell)$;
4 **end for**
5 let $B \overset{w,\varepsilon}{\leftarrow} A$ with $2\sigma_i^{B_1}(A) \le w_i \le 4\sigma_i^{B_1}(A)$;
6 **return** B

Lemma 3.6 (Repeated halving)**.** *Consider Algorithm 2. With high probability, every B_ℓ has $O(d \log(d))$ rows, and the output B has $O(d \log(d)/\varepsilon^2)$ rows and satisfies $B^T B \approx_\varepsilon A^T A$.*

Proof. First we prove that for every $1 \le \ell < L$, with high probability, B_ℓ has $O(d \log(d))$ rows and $B_\ell^T B_\ell \approx_{1/2} A_\ell^T A_\ell$. We prove this by induction. The base case $\ell = L$ is clear: $B_L = A_L$ and, by a multiplicative Chernoff bound[7], A_L has $\Theta(d)$ rows with high probability. For the inductive step, assume $B_{\ell+1}^T B_{\ell+1} \approx_{1/2} A_{\ell+1}^T A_{\ell+1}$. Then $\sigma_i^{B_{\ell+1}}(A_\ell) = (1 \pm 1/2)\sigma_i^{A_{\ell+1}}(A_\ell)$, so that the assumption on w_i implies that $\sigma_i^{A_{\ell+1}}(A_\ell) \le w_i \le 6\sigma_i^{A_{\ell+1}}(A_\ell)$. Hence we can apply Theorem 3.5 with $\varepsilon = 1/2$, which implies that with high probability $B_\ell^T B_\ell \approx_{1/2} A_\ell^T A_\ell$ and B_ℓ has $O(d \log d)$ rows.

It remains to prove that step 3. works correctly, assuming that $B_1 \approx_{1/2} A_1$, but this follows by the same argument. □

3.2 Quantum repeated halving algorithm

Here we discuss an efficient quantum algorithm for spectral approximation based on the repeated halving algorithm. This will prove Theorem 3.1. As we have seen, this requires efficient (quantum) algorithms for three steps: (i) a data structure that allows us to efficiently query approximate generalized leverage scores, (ii) a quantum implementation of the procedure $B \overset{w,\varepsilon}{\leftarrow} A$, and (iii) a way to query the downsampled matrices $A_L \subseteq_{1/2} \cdots \subseteq_{1/2} A_1 \subseteq_{1/2} A$. We discuss these three steps in the next three subsections and then combine them to obtain Theorem 3.1.

3.2.1 Leverage scores via Johnson-Lindenstrauss

Consider an approximation factor $\varepsilon > 0$. A bottleneck in the repeated halving algorithm is computing a multiplicative $(1 \pm \varepsilon)$-approximation of the generalized leverage scores $\sigma_i^B(A)$, where $A \in \mathbb{R}^{n \times d}$ has r_A-sparse rows a_i^T, and $B \in \mathbb{R}^{D \times d}$ has r_B-sparse rows with $D \ge d$. Naively (but exactly) computing the generalized leverage scores requires time $O(r_A^2)$ per leverage score, after a preprocessing cost of $O(\min\{Dd^{\omega-1}, Dr_B^2 + d^\omega\})$. Here the preprocessing consists of (i) computing the $d \times d$ matrix $B^T B$, which can be done in time $O(D/d \cdot d^\omega)$ by computing D/d matrix products of $d \times d$ matrices, or in time $O(Dr_B^2)$ by computing D outer products of r_B-sparse vectors, and (ii)

[7] Let $Y = \sum_{i=1}^n X_i$, with X_i the random variable indicating whether the i-th row of A is in A_L. Then $\mu := \mathbb{E}[Y] \in \Theta(d)$. The multiplicative Chernoff bound for random variables in $\{0,1\}$ states that $\Pr[|Y - \mu| \ge \delta\mu] \le 2e^{-\delta^2\mu/3}$ for $0 \le \delta \le 1$, and so $Y \in \Theta(d)$ except with probability $e^{-\Omega(d)}$.

computing the pseudo-inverse $(B^TB)^+$, which takes an additional time $O(d^\omega)$. From this we get the following lemma.

Lemma 3.7 (Direct leverage scores)**.** *Assume query access to $A \in \mathbb{R}^{n\times d}$, and assume we are explicitly given $B \in \mathbb{R}^{D\times d}$ with $n, D \gg d$. There is a classical algorithm that, after a precomputation of time $O(\min\{Dd^{\omega-1}, Dr_B^2 + d^\omega\})$, with high probability returns estimates $\tilde{\sigma}_i$ for any $i \in [n]$ that satisfy $\tilde{\sigma}_i = (1 \pm \varepsilon)\sigma_i^B(A)$ at a cost per estimate of one row query to A and time $O(r_A^2)$.*

Following a trick of Spielman and Srivastava [44], we can use the Johnson-Lindenstrauss lemma to get ε-approximate leverage scores at cost $O(r_A \log(n)/\varepsilon^2)$ per leverage score, after a similar precomputation cost. We will use the lemma in the following form.

Lemma 3.8 (Johnson-Lindenstrauss [23], [28, Lemma 1])**.** *For all integers $n, D \geq 0$ and precision $\varepsilon > 0$, there exists a distribution $\mathcal{D}_{n,D,\varepsilon}$ over matrices in $\mathbb{R}^{O(\log(n)/\varepsilon^2)\times D}$ such that the following holds: for any set of vectors $x_i \in \mathbb{R}^D$, $i \in [n]$, it holds that if $\Pi \sim \mathcal{D}_{n,D,\varepsilon}$ then with high probability*

$$\|\Pi x_i\|^2 = (1 \pm \varepsilon)\|x_i\|^2, \quad \forall i \in [n].$$

Moreover, the matrix $\Pi \sim \mathcal{D}_{n,D,\varepsilon}$ can be sampled in time $\widetilde{O}(D/\varepsilon^2)$.

To use this lemma, we start by rewriting the leverage scores as vector norms:

$$\sigma_i^B(A) = a_i^T(B^TB)^+a_i = a_i^T(B^TB)^+B^TB(B^TB)^+a_i = \left\|B(B^TB)^+a_i\right\|^2.$$

Now, by Lemma 3.8, we can sample a random matrix $\Pi \in \mathbb{R}^{O(\log(n)/\varepsilon^2)\times D}$ (independent of A, B) that with high probability (in n) satisfies

$$\left\|\Pi B(B^TB)^+a_i\right\|^2 = (1 \pm \varepsilon)\left\|B(B^TB)^+a_i\right\|^2, \quad \forall i \in [n].$$

Having already computed $(B^TB)^+$, we can compute the matrix $C = \Pi B(B^TB)^+ \in \mathbb{R}^{O(\log(n)/\varepsilon^2)\times d}$ in additional time $O(\frac{d\log(n)}{\varepsilon^2} \cdot (D + d))$: we first compute the $O(d\log(n)/\varepsilon^2)$ entries of ΠB, each of which is an inner product in dimension D, and then we do the same to compute $\Pi B(B^TB)^+$, this time with inner products in dimension d. After this we can return estimates $\tilde{\sigma}_i = \|Ca_i\|^2$ for the leverage scores, satisfying $\tilde{\sigma}_i = (1 \pm \varepsilon)\sigma_i^B(A)$, at a cost per estimate of 1 row-query to A and time $O(r_A \log(n)/\varepsilon^2)$. This leads to the following algorithm.

Algorithm 3: Generalized leverage scores via Johnson-Lindenstrauss:

Input: matrices $A \in \mathbb{R}^{n\times d}$, $B \in \mathbb{R}^{D\times d}$, approximation factor $\varepsilon > 0$

$-$ **Preprocessing:**
sample random matrix $\Pi \sim \mathcal{D}_{n,D,\varepsilon}$ and compute $C = \Pi B(B^TB)^+$;
sample random matrix $\Pi' \sim \mathcal{D}_{n,d,\varepsilon}$ and compute $C' = \Pi'(I - (B^TB)(B^TB)^+)$;
$-$ **Leverage score computation (index $i \in [n]$):**
compute $C'a_i$; if $C'a_i \neq 0$ then return $\tilde{\sigma}_i = \infty$;
otherwise, compute and return $\tilde{\sigma}_i = \|Ca_i\|^2$;

Based on this algorithm we can prove the following lemma.

Lemma 3.9 (Johnson-Lindenstrauss leverage scores)**.** *Assume query access to $A \in \mathbb{R}^{n\times d}$, and assume we are explicitly given an r_B-sparse matrix $B \in \mathbb{R}^{D\times d}$ with $n, D \gg d$. Algorithm 3, after a preprocessing of time $O(\min\{Dd^{\omega-1}, Dr_B^2 + d^\omega\} + \frac{dD\log(n)}{\varepsilon^2})$, returns with high probability estimates $\tilde{\sigma}_i$ for any $i \in [n]$ that satisfy $\tilde{\sigma}_i \approx_\varepsilon \sigma_i^B(A)$ at a cost per estimate of one row query to A and time $O(r_A \log(n)/\varepsilon^2)$.*

Proof. First we show correctness of the algorithm. By Lemma 3.8 we know that with high probability both $\|Ca_i\| = (1\pm\varepsilon)\|B(B^TB)^+a_i\|$ and $\|C'a_i\| = (1\pm\varepsilon)\|(I-(B^TB)(B^TB)^+)a_i\|$ for all $i \in [n]$. From $\|C'a_i\|$ we can now check whether $a_i \perp \ker(B)$ or not: indeed $I-(B^TB)(B^TB)^+$ corresponds to the projector onto $\ker(B)$, and so $a_i \perp \ker(B)$ iff $\|C'a_i\| = 0$. Hence, if $\|C'a_i\| = 0$ we can set $\tilde{\sigma}_i = \infty$ and otherwise we can set $\tilde{\sigma}_i = \|Ca_i\|^2$, which will be correct with high probability.

Now we prove the complexity bounds. First consider the precomputation phase. As discussed earlier, given B we can compute $(B^TB)^+$ and $(B^TB)(B^TB)^+$ in time $O(\min\{Dd^{\omega-1}, Dr_B^2 + d^\omega\})$, and $C = \Pi B(B^TB)^+$ and $C' = \Pi'(I-(B^TB)(B^TB)^+)$ in time $O(dD\log(n)/\varepsilon^2)$.

For the approximation of a single leverage score $\sigma_i^B(A)$, we query the entire row a_i of A, and compute $C'a_i$ and Ca_i in time $O(r_A\log(n)/\varepsilon^2)$. □

This improves on direct computation as in Lemma 3.7 when ε is not too small.

3.2.2 Grover search

The previous section discussed how we can efficiently access the sampling weights, which are based on leverage scores. Here we discuss how, given access to an appropriate weight vector $w \in (\mathbb{R}_{>0}\cup\{\infty\})^n$, we can sample a subsampled matrix $B \overset{w,\varepsilon}{\leftarrow} A$. For this we use a quantum sampling routine based on Grover search, as stated in the following lemma.

Lemma 3.10 ([5, Claim 3])**.** *Assume we have query access to a list $p = (p_i)_{i\in[n]} \in [0,1]^n$. There is a quantum algorithm that samples a subset $S \subseteq [n]$, such that $i \in S$ independently with probability p_i. The algorithm uses $\widetilde{O}(\sqrt{n\sum_i p_i})$ time and queries to p.*

From this, we can prove the following lemma on a quantum algorithm for implementing a single iteration of the repeated halving algorithm. By the repeated halving lemma (Lemma 3.6), with high probability, the returned matrix B will have $O(d\log(d)/\varepsilon^2)$ rows and satisfy $B \approx_\varepsilon A$.

Lemma 3.11 (Quantum halving algorithm)**.** *Assume query access to $A \in \mathbb{R}^{n\times d}$ with row sparsity r. Consider a random submatrix $A' \subseteq_{1/2} A$ and assume that we are given $B' \in \mathbb{R}^{O(d\log d)\times d}$, with row sparsity r, such that $B'^TB' \approx_{1/2} A'^TA'$. For $\varepsilon \in (0,1]$, there is a quantum algorithm that with high probability returns a matrix $B \overset{w,\varepsilon}{\leftarrow} A$ for $w_i = \min\{1, \tilde{w}_i\}$ with $2\sigma_i^{B'}(A) \le \tilde{w}_i \le 4\sigma_i^{B'}(A)$. In expectation, the algorithm takes $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to A and $\widetilde{O}(r\sqrt{nd}/\varepsilon + d^\omega)$ time.*

Proof. First we invoke Lemma 3.9 on A and B', with $D = O(d\log(d))$. It states that, after a precomputation time of $\widetilde{O}(d^\omega)$, we can query approximate leverage scores $\tilde{\sigma}_i$ satisfying $\tilde{\sigma}_i \approx_{1/2} \sigma_i^{B'}(A)$ for $i \in [n]$, by querying one row of A and using time $O(r_A\log(n))$ per leverage score. Now set $w_i = \min\{1, \tilde{w}_i\}$ with $\tilde{w}_i = 2\tilde{\sigma}_i$, which satisfies $2\sigma_i^{B'}(A) \le \tilde{w}_i \le 4\sigma_i^{B'}(A)$. We define the list $p = (p_i)_{i\in[n]}$ by setting $p_i = \min\{1, cw_i\log(d)/\varepsilon^2\}$ (as in Section 3.1), and so we can query p_i at the same cost of querying $\tilde{\sigma}_i$.

To sample $B \overset{w,\varepsilon}{\leftarrow} A$, we apply the algorithm from Lemma 3.10 to the list p. Theorem 3.5 implies that $\sum_i p_i \in O(d \log(d)/\varepsilon^2)$, and so the complexity amounts to $\widetilde{O}(\sqrt{nd}/\varepsilon)$ queries to p. Combined with the previous paragraph, this gives a total complexity of $\widetilde{O}(r_A\sqrt{nd}/\varepsilon + d^\omega)$ time and $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to A. □

3.2.3 Random oracle access

To obtain our final quantum algorithm we would like to repeatedly apply Lemma 3.11 in the repeated halving algorithm. This is plug-and-play, up to one detail, which is how to query the downsampled matrices $A_L \subseteq_{1/2} \cdots \subseteq_{1/2} A_1 \subseteq_{1/2} A$. Notice that these can be defined implicitly by drawing L bitstrings $\{X^{(\ell)} \in \{0,1\}^n\}_{\ell \in [L]}$ with $X_i^{(\ell)} = 1$ independently with probability $1/2$. The matrix A_k then contains the i-th row of A if and only if $\Pi_{\ell \le k} X_i^{(\ell)} = 1$. Hence, if we could efficiently query such a "random oracle" then we're done.

Now, if we only care about query complexity, then we can sample these bitstrings explicitly and use QRAM access to them. However, if we want to have a sublinear runtime as well, then explicitly drawing these bitstrings would be too expensive (scaling as $\Omega(n)$). We remedy this by invoking the following lemma, stating that we can efficiently *simulate* access to a random oracle.

Lemma 3.12 (Quantum random oracles [5, Claim 1])**.** *Consider any quantum algorithm with runtime q that makes queries to a uniformly random string. We can simulate this algorithm with a quantum algorithm with runtime $\widetilde{O}(q)$ without random string, using an additional QRAM of $\widetilde{O}(q)$ bits.*

3.2.4 Quantum repeated halving algorithm

We state our final algorithm below.

Algorithm 4: Quantum repeated halving algorithm:

Input: query access to $A \in \mathbb{R}^{n \times d}$, approximation factor $\varepsilon > 0$

1 implicitly construct the chain $A_L \subseteq_{1/2} \cdots \subseteq_{1/2} A_1 \subseteq_{1/2} A$ for $L = O(\log(n/d))$;
2 use Grover search[8] to explicitly learn A_L; let $B_L = A_L$;
3 **for** $\ell = L-1, L-2, \ldots, 1$ **do**
4 use Lemma 3.11 on $A_{\ell+1} \subseteq_{1/2} A_\ell$ and $B_{\ell+1}$ to explicitly construct $B_\ell \overset{w,1/2}{\leftarrow} A_\ell$ with $w_i = \min\{1, \tilde{w}_i\}$ and $2\sigma_i^{B_{\ell+1}}(A_\ell) \le \tilde{w}_i \le 4\sigma_i^{B_{\ell+1}}(A_\ell)$;
5 **end for**
6 use Lemma 3.11 on $A_1 \subseteq_{1/2} A$ and B_1 to explicitly construct $B \overset{w,\varepsilon}{\leftarrow} A$ with $w_i = \min\{1, \tilde{w}_i\}$ and $2\sigma_i^{B_1}(A) \le \tilde{w}_i \le 4\sigma_i^{B_1}(A)$;
7 **return** B

Based on this algorithm, we can prove the following theorem, which implies our main theorem on quantum spectral approximation (Theorem 3.1).

[8] More specifically, apply Lemma 3.10 with $p_i = \Pi_{\ell \le L} X_i^{(\ell)}$, where the $X_i^{(\ell)}$'s were defined in Section 3.2.3.

Theorem 3.13 (Quantum repeated halving algorithm)**.** *With high probability, the quantum repeated halving algorithm (Algorithm 4) returns a matrix B with $O(d\log(d)/\varepsilon^2)$ rows that satisfies $B^TB \approx_\varepsilon A^TA$. The algorithm makes $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to A and takes time $\widetilde{O}(d^\omega + r\sqrt{nd}/\varepsilon)$.*

Proof. Correctness of the algorithm directly follows from correctness of the original repeated halving algorithm (Lemma 3.6). It remains to prove the query and runtime guarantees. By the discussion in Section 3.2.3, we can assume efficient access to a random oracle to query the A_ℓ matrices in Step 1. We use Lemma 3.10 to implement Step 2.: we learn the $\Theta(d)$ row indices of A_L among A's n rows in time $\widetilde{O}(\sqrt{nd})$, and we explicitly learn the corresponding rows by making $\Theta(d)$ row queries to A. For Step 3. and 4., we repeatedly apply Lemma 3.11, requiring a total time of $\widetilde{O}(d^\omega + r_A\sqrt{nd}/\varepsilon)$ and $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to A. □

Based on this we can also prove the theorem about approximating leverage scores, which we restate below for convenience.

Theorem 3.2 (Quantum leverage score approximation)**.** *Assume query access to a matrix $A \in \mathbb{R}^{n\times d}$ with row sparsity r. For any $0 < \varepsilon \leq 1$, there is a quantum algorithm that, with high probability, provides query access to estimates $\tilde{\sigma}_i$ for any $i \in [n]$ satisfying*

$$\tilde{\sigma}_i = (1 \pm \varepsilon)\sigma(A).$$

The cost of the algorithm is either of the following:

1. *Preprocessing: $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to A and $\widetilde{O}(r\sqrt{nd}/\varepsilon + d^\omega + \min\{d^\omega, dr^2/\varepsilon^2\}/\varepsilon^2)$ time. Cost per estimate $\tilde{\sigma}_i$: 1 row query to A and $O(r^2)$ time.*
2. *Preprocessing: $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to A and $\widetilde{O}(r\sqrt{nd}/\varepsilon + d^\omega + \min\{d^\omega, dr^2/\varepsilon^2\}/\varepsilon^2 + d^2/\varepsilon^4)$ time. Cost per estimate $\tilde{\sigma}_i$: 1 row query to A and $\widetilde{O}(r/\varepsilon^2)$ time.*

Proof. The quantum algorithm first computes $B \in \mathbb{R}^{d\log(d)/\varepsilon^2 \times d}$ such that $B^TB \approx_{\varepsilon/2} A^TA$, with high probability. By Theorem 3.13 this takes $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to A and $\widetilde{O}(d^\omega + r_A\sqrt{nd}/\varepsilon)$ time. Now note that $B^TB \approx_{\varepsilon/2} A^TA$ implies that $\sigma_i^B(A) \approx_{\varepsilon/2} \sigma_i(A)$, and so we can invoke either Lemma 3.7 or Lemma 3.9 to approximate $\sigma_i(A)$, with $D \in \widetilde{O}(d/\varepsilon^2)$ and $r_B = r_A = r$. From Lemma 3.7 we get an additional preprocessing cost of $\widetilde{O}(d^\omega + \min\{d^\omega, dr^2\}/\varepsilon^2)$ and a cost per estimate of 1 row query to A and time $O(r^2)$. From Lemma 3.9 we get an additional preprocessing cost of $\widetilde{O}(d^\omega + \min\{d^\omega, dr^2\}/\varepsilon^2 + d^2/\varepsilon^4)$ and a cost per estimate of 1 row query to A and time $\widetilde{O}(r/\varepsilon^2)$. □

4 Quantum algorithm for ℓ_p-Lewis weights

In this section we describe our quantum algorithm for approximating Lewis weights. The algorithm builds on our quantum algorithms for spectral approximation and leverage scores computation. It combines a stability lemma from [21] with the approximate Lewis weight algorithm from [30] to obtain $(1 \pm \varepsilon)$-multiplicative estimates of ℓ_p-Lewis weights using $\widetilde{O}(\sqrt{d}/\varepsilon)$ leverage score computations, each with precision $\approx \varepsilon/\sqrt{d}$. This suffices for the Lewis weight IPM, but it gives a high poly(d)-dependence.

Let us first recall some definitions. For $p > 0$ the ℓ_p-Lewis weights $w^{(p)}(A) \in \mathbb{R}^n_{>0}$ of a matrix $A \in \mathbb{R}^{n\times d}$ are defined as the unique vector $w \in \mathbb{R}^n_{>0}$ such that

$$w_i = \sigma_i(W^{\frac{1}{2}-\frac{1}{p}}A) = w_i^{1-\frac{2}{p}} a_i^T (A^T W^{1-\frac{2}{p}} A)^+ a_i, \quad \text{for all } i \in [n],$$

where $W = \mathrm{Diag}(w)$. The i-th Lewis weight w_i is hence defined implicitly as the i-th leverage score of the matrix $W^{\frac{1}{2}-\frac{1}{p}}A$. This suggests the following notion of Lewis weight approximation, which seems to be the easiest to handle.

Definition 4.1 (ε-FP-approximate ℓ_p-Lewis weights [30])**.** *For a matrix $A \in \mathbb{R}^{n\times d}$ we call a vector $v \in \mathbb{R}^n$ an ε-fixed-point-approximate (ε-FP-approximate) ℓ_p-Lewis weight for A if*

$$v_i \approx_\varepsilon \sigma_i(V^{\frac{1}{2}-\frac{1}{p}}A) \qquad \text{for all } i \in [n], \tag{4}$$

where $V = \mathrm{Diag}(v)$.

In their thesis, Lee [30] shows how to compute such ε-FP-approximate Lewis weights using $\widetilde{O}(1/\varepsilon)$ leverage score computations, see Theorem 4.2 and Algorithm 5 below.

Theorem 4.2 ([30, Thrm. 5.3.4])**.** *For any matrix $A \in \mathbb{R}^{n\times d}$, accuracy $0 < \varepsilon < 1$ and $p \geq 2$, Algorithm 5 outputs ε-FP-approximate ℓ_p-Lewis weights for A. That is, the output $\overline{v}$ satisfies*

$$\overline{v} \approx_\varepsilon \sigma(\overline{V}^{\frac{1}{2}-\frac{1}{p}}A),$$

where $\overline{V} = \mathrm{Diag}(\overline{v})$.

Algorithm 5: Approximate ℓ_p-Lewis weights

Input: $A \in \mathbb{R}^{n\times d}$, accuracy $0 < \varepsilon < 1$, $p \geq 2$
Output: A vector $\overline{v} \in \mathbb{R}^n$

Let $v_i^{(1)} = d/n$ for all $i \in [n]$ and set $T = 2\log(n/d)/\varepsilon$;
for $k = 1, \ldots, T$ **do**
  Let $\tilde{\sigma}$ be an entrywise multiplicative $(1 \pm \varepsilon/4)$-approximation of $\sigma((V^{(k)})^{\frac{1}{2}-\frac{1}{p}}A)$;
  Let $v^{(k+1)} = \tilde{\sigma}$ and $V^{(k+1)} = \mathrm{Diag}(v^{(k+1)})$;
end for
return $\overline{v} = \frac{1}{T}\sum_{k=1}^T v^{(k)}$

4.1 Iterative quantum algorithm for approximate Lewis weights

We obtain a quantum algorithm for computing Lewis weights by plugging our quantum algorithm for computing leverage scores (Theorem 3.13) into Lee's algorithm. We have to be careful because in sublinear time we cannot afford to explicitly write down $\Omega(n)$ coefficients, and so we can only *implicitly* keep track of the coefficients throughout the algorithm.

Theorem 4.3 (Quantum Lewis weight approximation)**.** *Consider query access to a matrix $A \in \mathbb{R}^{n\times d}$ with row sparsity r and $n \gg d$. For any $0 < \varepsilon \leq 1$ and $p \geq 2$, there is a quantum algorithm that, with*

high probability, provides query access to a vector $\overline{v} \in \mathbb{R}^n$ of ε-FP-approximate ℓ_p-Lewis weights. I.e.,

$$\overline{v} \approx_\varepsilon \sigma(\overline{V}^{\frac{1}{2}-\frac{1}{p}} A),$$

where $\overline{V} = \mathrm{Diag}(\overline{v})$. The cost of the algorithm is either of the following:

1. *(Direct approach)*
   Preprocessing: $\widetilde{O}(\sqrt{nd}/\varepsilon^2)$ *row queries to A and* $\widetilde{O}(\sqrt{nd}r^2/\varepsilon^3 + d^\omega/\varepsilon + \min\{d^\omega, dr^2\}/\varepsilon^3)$ *time.* Cost per estimate $\overline{v}_i$: *1 row query to A and $O(r^2/\varepsilon)$ time.*
2. *(Johnson-Lindenstrauss approach)*
   Preprocessing: $\widetilde{O}(\sqrt{nd}/\varepsilon^2)$ *row queries to A and* $\widetilde{O}(\sqrt{nd}r/\varepsilon^5 + d^\omega/\varepsilon + \min\{d^\omega, dr^2\}/\varepsilon^3 + d^2/\varepsilon^5)$ *time.* Cost per estimate $\overline{v}_i$: *1 row query to A and $\widetilde{O}(r/\varepsilon^3)$ time.*

Proof. We establish the theorem by invoking Theorem 3.2 inside Lee's algorithm. Let $A^{(k)} = (V^{(k)})^{1/2-1/p} A$. The theorem bounds quantities p_q, p_t and q_t so that we can simulate query access to $v_i^{(k+1)} \approx_{\varepsilon/4} \sigma(A^{(k)})$ with the following costs:[9]

1. *Preprocessing:* p_q row queries to A and $v^{(k)}$, and p_t time.
2. *Cost per query to $v_i^{(k+1)}$:* 1 row query to i-th row of A and $v^{(k)}$, and q_t time.

Specifically, Theorem 3.2 gives bounds $p_q \in \widetilde{O}(\sqrt{nd}/\varepsilon)$ and either (i) $p_t \in \widetilde{O}(\min\{d^\omega/\varepsilon^2, d^\omega + dr^2/\varepsilon^2\} + r\sqrt{nd}/\varepsilon)$ and $q_t \in O(r^2)$, or (ii) $p_t \in \widetilde{O}(\min\{d^\omega/\varepsilon^2, d^\omega + dr^2/\varepsilon^2\} + r\sqrt{nd}/\varepsilon + d^2/\varepsilon^4)$ and $q_t \in \widetilde{O}(r/\varepsilon^2)$.

Bounding the cost of implementing Lee's algorithm requires some care. For starters, we bound the query and time complexity of a single query to $v_i^{(k)}$ or $\bar{v}_i$ *after preprocessing*. A query to $v_i^{(k)}$ can be simulated with 1 row query to the i-th row of A, 1 query to $v_i^{(k-1)}$, and time q_t. On its turn, $v_i^{(k-1)}$ can be queried with 1 row query to the i-th row of A (which we already queried!), 1 query to $v_i^{(k-2)}$, and time q_t. Repeating this argument, and noting that $v_i^{(1)} = n/d$, it follows that $v_i^{(k)}$ can be queried with 1 row query to the i-th row of A and total time $O(kq_t)$. To query $\bar{v}_i = \frac{1}{T}\sum_{k=1}^{T} v_i^{(k)}$, we note that it is not more expensive then the cost to query $v_i^{(T)}$, since the latter effectively queries each of $v^{(1)}, \ldots, v^{(T-1)}$. Hence we can query $\overline{v}_i$ also using 1 row-query to A and time $O(Tq_t)$.

Now we bound the preprocessing cost per iteration. We solve the iterations chronologically, so that in iteration $k+1$ we can assume that the preprocessing is done for previous iterations. That said, in iteration $k+1$ we have to make p_q row queries to A and $v^{(k)}$, and spend time p_t. By the preceding argument, we can make p_q queries to $v^{(k)}$ by making p_q row queries to A and spending time $O(p_q k q_t)$, so that in total we make $2p_q$ row queries to A and spend time $O(p_t + p_q k q_t)$. Summing over all $k = 1, 2, \ldots, T$ iterations, we get a total row query complexity of $O(Tp_q)$ and time complexity of $O(Tp_t + T^2 p_q q_t)$.

Recalling that $T \in \widetilde{O}(1/\varepsilon)$, and filling in the bounds on p_q, p_t and q_t from Theorem 3.2, we proved the claimed complexities. □

[9] We use that a row query to the i-th row of $A^{(k)}$ can be simulated by 1 row query to the i-th row of A and 1 query to $v_i^{(k)}$.

4.2 Stability of Lewis weights

While the notion of ε-FP-approximation (Definition 4.1) is very natural when approximating Lewis weights, their algorithmic usage (in the context of IPMs) seems easiest when we are given the usual ε-multiplicative approximation. Unfortunately, Lewis weights are known to be unstable with respect to perturbation (for $p \geq 4$, which we assume throughout this section), and this seems to make their multiplicative approximation much harder (see [21] for some recent advances). In that same work, Fazel, Lee, Padmanabhan and Sidford [21] prove a notion of stability that allows one to retrieve an ε-multiplicative approximation of the ℓ_p-Lewis weights from an $O(\varepsilon/(p^2\sqrt{d}))$-FP-approximation.

To see this, we reformulate the notion of ε-FP-approximation in the language of [21]. First, note that Eq. (4) is equivalent to

$$\frac{1}{1+\varepsilon} \leq \frac{a_i^T (A^T V^{1-\frac{2}{p}} A)^+ a_i}{v_i^{2/p}} \leq \frac{1}{1-\varepsilon} \qquad \text{for all } i \in [n].$$

Now let $\hat{v} = v^{1-\frac{2}{p}} = v^{\frac{p-2}{p}}$, so that $v^{2/p} = \hat{v}^{\frac{2}{p-2}} = \hat{v}^{\alpha}$ where $\alpha := \frac{2}{p-2}$. Then we have

$$\frac{a_i^T (A^T V^{1-\frac{2}{p}} A)^+ a_i}{v_i^{2/p}} = \frac{a_i^T (A^T \widehat{V} A)^+ a_i}{\hat{v}_i^{\alpha}}.$$

Following [21], for a vector $v \in \mathbb{R}^n_{>0}$, let us write

$$\rho_i^{(p,A)}(v) := \frac{\sigma_i(V^{1/2}A)}{v_i^{1+\alpha}} = \frac{a_i^T (A^T V A)^+ a_i}{v_i^{\alpha}}, \qquad \text{for all } i \in [n].$$

Clearly, if v is a vector of ε-FP-approximate ℓ_p-Lewis weights of a matrix A, then $\rho^{(p,A)}(\hat{v})$ is coordinate-wise close to the all-ones vector. We can now use the following stability property.

Lemma 4.4 ([21, Lem. 14])**.** *Consider an integer $p \geq 4$ and matrix $A \in \mathbb{R}^{n\times d}$. Let $\hat{v} \in \mathbb{R}^n_{>0}$ be a vector such that*

$$\exp(-\mu) \leq \rho_i^{(p,A)}(\hat{v}) \leq \exp(\mu), \qquad \textit{for all } i \in [n].$$

Then $\hat{v}$ approximates the ℓ_p-Lewis weights $w^{(p)}(A)$ in the following sense (recall $\alpha = 2/(p-2)$):

$$\exp\left(-\frac{1}{\alpha}(1+\sqrt{d}/\alpha)\mu\right) w_i^{(p)}(A) \leq \hat{v}_i \leq \exp\left(\frac{1}{\alpha}(1+\sqrt{d}/\alpha)\mu\right) w_i^{(p)}(A).$$

Note that for $p \geq 4$ we have $\alpha \leq 1$ and thus the dominating term in the exponent of the approximation factor is $\sqrt{d}\mu/\alpha^2$. In other words, if we set $\mu = \Theta(\varepsilon\alpha^2/\sqrt{d}) = \Theta(\varepsilon/(p^2\sqrt{d}))$ and we use that $\frac{1}{1-\varepsilon} \approx \exp(\varepsilon)$ for small ε, then we obtain an ε-multiplicative approximation of the true ℓ_p-Lewis weights from a set of μ-FP-approximate Lewis weights. Combining this observation with our FP-approximate Lewis weights algorithm (Theorem 4.3) gives the following theorem.

Theorem 4.5 (Multiplicative approximations of Lewis weights)**.** *Assume quantum query access to $A \in \mathbb{R}^{n\times d}$. Let $0 < \varepsilon < 1$ and $p \geq 4$ be given. There is a quantum algorithm that sets up quantum query access to a vector $\overline{v} \in \mathbb{R}^n$ that satisfies $\overline{v} \approx_\varepsilon w^{(p)}(A)$. The algorithm has a preprocessing cost of $\widetilde{O}(\sqrt{n}d^{3/2}/\varepsilon^2)$ row queries to A and time $\widetilde{O}(\sqrt{n}d^2r^2/\varepsilon^3 + d^{\omega+1/2}/\varepsilon + \min\{d^{\omega+3/2}, d^{5/2}r^2\}/\varepsilon^3)$. Each query to $\overline{v}$ then requires 1 row-query to A and time $\widetilde{O}(r^2\sqrt{d}/\varepsilon)$.*

5 Approximate matrix-vector product

In this section we describe our quantum algorithm for approximation matrix-vector products. It combines the quantum algorithm for spectral approximation (Theorem 3.1) with quantum multivariate mean estimation [19]. A blueprint of the algorithm is given in Algorithm 6. As explained in the introduction, for our Newton step we wish to approximate a matrix-vector product $y = Bv$ in the "local inverse norm", i.e., we wish to return $\tilde{y}$ such that $\|\tilde{y} - y\|_{W^{-1}} \leq \delta$ with $W = B^T B$. Since the algorithm from [19] only gives an approximation in the 2-norm, we first use a spectral approximation $\widetilde{W} \approx W$ to "precondition" the product to $z = \widetilde{W}^{-1/2} y$. A 2-norm approximation of z then translates to a local norm approximation of y:

$$\|\tilde{z} - z\|_2 = \|\widetilde{W}^{1/2}\tilde{z} - y\|_{\widetilde{W}^{-1}} \approx \|\widetilde{W}^{1/2}\tilde{z} - y\|_{W^{-1}},$$

so that we can return $\tilde{y} = \widetilde{W}^{1/2}\tilde{z}$.

Algorithm 6: Quantum algorithm to approximate matrix-vector products

Input: $B \in \mathbb{R}^{n\times d}$, $v \in \mathbb{R}^n$, bound $\alpha \geq \|v\|_\infty$, accuracy $0 < \delta < 1$
Output: A vector $\tilde{y} \in \mathbb{R}^d$

1 Compute $\widetilde{W} \approx_{1/2} B^T B$ using quantum spectral approximation (Theorem 3.1);
2 Compute $\widetilde{W}^{-1/2}$ classically;
3 Define the random variable $X = -nv_\ell \widetilde{W}^{-1/2} B^T |\ell\rangle$, with $\ell \in [n]$ uniformly at random;
4 Use quantum multivariate mean estimation on X with $T = \widetilde{O}(\sqrt{n}\alpha d/\delta)$ quantum samples to obtain $\tilde{\mu} \in \mathbb{R}^d$ such that $\|\tilde{\mu} - \mathbb{E}[X]\|_2 \leq \delta$;
5 **return** $\tilde{y} = \widetilde{W}^{1/2}\tilde{\mu}$

Theorem 5.1 (Approximate matrix-vector product)**.** *Assume query access to a vector $v \in \mathbb{R}^n$ and a matrix $B \in \mathbb{R}^{n\times d}$ with row sparsity r. There is a quantum algorithm that returns a vector $\tilde{y}$ satisfying*

$$\|\tilde{y} - B^T v\|_{(B^T B)^{-1}} \leq \delta,$$

while making $\widetilde{O}(\sqrt{n}d\|v\|_\infty/\delta)$ row queries to B and v, and taking time $\widetilde{O}(r\sqrt{n}d^2\|v\|_\infty/\delta + d^\omega)$.

Proof. Let $W = B^T B$ and $y = B^T v$. Let $\gamma = \frac{1}{2}$. First, we use Theorem 3.1 to construct a spectral approximation $\widetilde{W} \approx_\gamma W$ with $\widetilde{O}(r\sqrt{nd})$ quantum queries and time $\widetilde{O}(r\sqrt{nd} + d^\omega)$. We then classically compute $\widetilde{W}^{-1/2}$, which takes time $\widetilde{O}(d^\omega)$.

Now, define the d-dimensional random variable

$$X = -nv_\ell \widetilde{W}^{-1/2} B^T |\ell\rangle,$$

with $\ell \in [n]$ uniformly at random. Then X has mean

$$\mu = \mathbb{E}[X] = -\frac{1}{n}\sum_{\ell=1}^{n} nv_\ell \widetilde{W}^{-1/2} B^T |\ell\rangle = \widetilde{W}^{-1/2} B^T v.$$

To upper bound the trace of the covariance matrix Σ, note that

$$\begin{aligned}\sigma_i^2 = \mathbb{E}[X_i^2] - \mathbb{E}[X_i]^2 \leq \mathbb{E}[X_i^2] &= n \langle i|\widetilde{W}^{-1/2} B^T V^2 B \widetilde{W}^{-1/2}|i\rangle \\ &\leq n\|v\|_\infty^2 \langle i|\widetilde{W}^{-1/2} W \widetilde{W}^{-1/2}|i\rangle \leq (1+\gamma) n \|v\|_\infty^2,\end{aligned}$$

where we used that $\tilde{W}^{-1/2} W \tilde{W}^{-1/2} \approx_\gamma I$. This implies that $\mathrm{Tr}(\Sigma) = \sum_{i=1}^d \sigma_i^2 \leq (1+\gamma)\|v\|_\infty^2 nd$.

Now we can use quantum multivariate mean estimation [19, Thrm. 3.5]. We can get an estimate $\tilde{\mu}$ satisfying

$$\|\tilde{\mu} - \mu\|_2 \leq \delta/2$$

using $T = \widetilde{O}(\sqrt{d\,\mathrm{Tr}(\Sigma)}/\delta) \leq \widetilde{O}(\sqrt{\|v\|_\infty^2 nd}/\delta)$ quantum samples of X. Finally, we return the estimate $\tilde{y} = \widetilde{W}^{1/2}\tilde{\mu}$. This estimate satisfies

$$\|\tilde{y} - y\|_{W^{-1}} \leq (1+\gamma)\|\tilde{y} - y\|_{\tilde{W}^{-1}} = (1+\gamma)\|\tilde{\mu} - \mu\|_2 \leq (1+\gamma)\delta/2 \leq \delta.$$

The total sample complexity is $T \in \widetilde{O}(\sqrt{n}d\|v\|_\infty/\delta)$ and we can obtain a sample from X using one row query to B, one query to v, and time dr (recall that we already computed $\widetilde{W}^{-1/2}$). The total time complexity is hence $\widetilde{O}(r\sqrt{n}d^2\|v\|_\infty/\delta)$.[10] □

Remark 5.2. *While we omit an explicit treatment, one can also use* univariate *quantum mean estimation [38] to estimate μ coordinate-wise. While this improves the time complexity by a factor $\sqrt{d}$, it increases the number of queries by a factor $\sqrt{d}$.*

6 Robustness of IPMs with respect to approximating the Newton step

In this section we describe a robust version of an interior point method for convex optimization that is amenable to a quantum speedup. More specifically, we present a variation of the "short-step barrier method" that replaces the Newton step by an approximation of it. We show that such approximate steps still allow us to follow the central path. This leads to the following theorem. The argument is relatively standard, but since we didn't find an appropriate reference we provide it in Appendix B. We defer the definitions of *self-concordance* and *complexity* of a barrier to Section 6.1.1, for now we just mention that all barrier functions that we consider are self-concordant and have known complexity parameters.

Theorem 6.1 (IPM based on approximate Newton steps)**.** *Consider the convex optimization problem $\min_{x\in D} c^T x$ for convex region D, and let $f : D \to \mathbb{R}$ be a self-concordant barrier function for D with complexity ϑ_f, gradient $g(x)$ and Hessian $H(x)$. Assume that we are given an appropriate initial point $z_0 \in D$ (see Remark 6.2). Then, for any $\varepsilon > 0$, we give an IPM algorithm that returns $x' \in D$ satisfying $c^T x' \leq \mathrm{val} + \varepsilon$. The algorithm computes $O(\sqrt{\vartheta_f}\log(1/\varepsilon))$ many approximate gradients $\tilde{g}(x)$ and Hessians $\tilde{H}(x)$ satisfying*

$$\|\tilde{g}(x) - g(x)\|_{H(x)} \leq 1/\operatorname{polylog}(n) \quad \text{and} \quad \tilde{H}(x) \preceq H(x) \preceq \operatorname{polylog}(n)\,\tilde{H}(x).$$

[10] Ref. [19] does not explicitly state the time complexity of their algorithm, but their overhead is essentially due to the quantum Fourier transform in dimension d and, per quantum sample, the computation of d-dimensional inner products. This overhead is less than the time dr that we spend to prepare a quantum sample.

We will instantiate the above for linear programs with the logarithmic barrier, the volumetric barrier and the Lewis weight barrier. At a first read, readers can treat this theorem as a black-box and proceed with Section 7 for quantum implementations of such approximations.

Remark 6.2 (Initial point)**.** *As is customary for interior point methods, we assume that we are given some initial point x_0 in the interior of the feasible region D. For Theorem 6.1 we assume that there exists a small but constant $\eta > 0$ such that z_0 is an approximate minimizer of $f_\eta(x) = \eta c^T x + f(x)$. More specifically, we require that the Newton step at z_0 is small in the local norm (see later for definitions). By a standard argument of "path-switching" (see, e.g., [42, Sec. 2.4.2]) this assumption is equivalent to the assumption that we are given an initial point x_0 in the interior of D that is "not too close to the boundary" (e.g., it suffices to have $B(x_0, 1/\operatorname{poly}(n,d) \subseteq D \subseteq B(x_0, \operatorname{poly}(n,d))$ in the Euclidean norm).*

6.1 Self-concordant barriers and path-following

For this section we follow the exposition of [42], but similar statements can be found e.g. in [39].

We consider an optimization problem of the form

$$\min_{x \in D} c^T x$$

for some open, bounded, full-dimensional, convex region $D \subset \mathbb{R}^d$ and a cost function $c \in \mathbb{R}^d$. Let val denote the optimum. We turn this into an unconstrained optimization problem by using a *barrier function* $f : D \to \mathbb{R}$ that satisfies $f(x) \to \infty$ as x approaches the boundary of D. We assume that f is twice continuously differentiable with gradient $g(x)$ and positive definite Hessian $H(x)$.

We first recall that the Hessian defines a "local" inner product by

$$\langle a, b \rangle_{H(x)} := \langle a, b \rangle_x := a^T H(x) v$$

for $a, b \in \mathbb{R}^d$. In turn this defines the local norm $\|v\|_{H(x)} := \|v\|_x := \sqrt{v^T H(x) v}$. Note that the gradient $g(x)$ and Hessian $H(x)$ are defined with respect to the standard inner product. We will use $g_x(y) := H(x)^{-1} g(y)$ and $H_x(y) = H(x)^{-1} H(y)$ to denote the gradient and Hessian at y with respect to the local inner product at x. In particular, note that $g_x(x) = H^{-1}(x) g(x) =: -n(x)$ is the Newton step at x and $H_x(x) = I$.

6.1.1 Self-concordance

As an additional requirement, we impose that f is "self-concordant". In the following, let $B_x(y, r)$ denote the open ball of radius r centered at y in the norm $\|\cdot\|_x$.

Definition 6.3 (Self-concordance)**.** *The function f is* self-concordant[11] *if for all $x \in D$ we have $B_x(x, 1) \subseteq D$, and if whenever $y \in B_x(x, 1)$ we have*

$$1 - \|y - x\|_x \leq \frac{\|v\|_y}{\|v\|_x} \leq \frac{1}{1 - \|y - x\|_x} \quad \text{for all } v \neq 0.$$

[11] Technically speaking, we assume strongly nondegenerate self-concordant, where the term "strongly" refers to the requirement $B_x(x, 1) \subseteq D$, and "nondegenerate" refers to a positive definite Hessian at each x (and hence a local inner product at each x).

For example, if in the above Definition 6.3 we have $\|y - x\|_x \leq 1/2$, then $\frac{1}{2}\|v\|_x \leq \|v\|_y \leq 2\|v\|_x$ for all v.

The self-concordance condition is roughly the same as Lipschitzness of the Hessian of f, in the following way. (Note that $\frac{1}{(1-\|y-x\|_x)^2} - 1 \approx 2\|y - x\|_x$ for small $\|y - x\|_x$, and $H_x(x) = I$.)

Theorem 6.4 ([42, Thrm. 2.2.1])**.** *Assume f has the property that $B_x(x, 1) \subseteq D$ for all $x \in D$. Then f is self-concordant if and only if for all $x \in D$ and $y \in B_x(x, 1)$ we have*

$$\|H_x(y)\|_x, \|H_x(y)^{-1}\|_x \leq \frac{1}{(1 - \|y - x\|_x)^2}; \tag{5}$$

likewise, f is self-concordant if and only if

$$\|I - H_x(y)\|_x, \|I - H_x(y)^{-1}\|_x \leq \frac{1}{(1 - \|y - x\|_x)^2} - 1. \tag{6}$$

A last quantity that we associate to a barrier function is its *complexity*, which is defined below. We use it to bound the complexity of path-following in the next section.

Definition 6.5 (Barrier complexity)**.** *A functional f is called a* (strongly nondegenerate self-concordant) barrier *if f is self-concordant and*

$$\vartheta_f := \sup_{x \in D} \|g_x(x)\|_x^2 < \infty.$$

The value ϑ_f is called the complexity *of f.*

6.1.2 Path-following

In an IPM we use the barrier function f to encode the convex set D, and we solve the unconstrained problem $\min_x f_\eta(x)$ where

$$f_\eta(x) := \eta c^T x + f(x),$$

for $\eta > 0$ and f a barrier function. We let $z(\eta)$ denote the minimizer of f_η, and call $\{z(\eta)\}_{\eta > 0}$ the *central path*. If f is self-concordant with complexity ϑ_f, then we have the following two properties, see Eqs. (2.12) and (2.13) in [42].

1. As $\eta \to \infty$, $z(\eta)$ approaches the minimizer of $\langle c, x \rangle$ over D:

$$c^T z(\eta) \leq \text{val} + \frac{1}{\eta}\vartheta_f.$$

2. Moreover, for $y \in B_{z(\eta)}(z(\eta), 1)$ we have

$$c^T y \leq \text{val} + \frac{1}{\eta}\vartheta_f + \frac{1}{\eta}\vartheta_f \|y - z(\eta)\|_{z(\eta)} \leq \text{val} + \frac{2}{\eta}\vartheta_f.$$

The idea of *path-following* is then to start from a point close to the central path for some η and alternate the following two steps: (i) increase $\eta \to \eta'$, and (ii) move the current iterate x close to $z(\eta')$. At a high level, if we increase η by a small enough amount, then the current iterate x is already close enough to $z(\eta')$ to ensure that Newton's method brings us very close to $z(\eta')$ with few iterations. Once η is roughly ϑ_f/ε, any point at a constant distance from $z(\eta)$ is ε-close to optimal; one can show that updating η multiplicatively by a factor roughly $1 + 1/\sqrt{\vartheta_f}$ is a safe choice and hence roughly $\widetilde{O}(\sqrt{\vartheta_f}\log(\vartheta_f/\varepsilon))$ iterations suffice.

Traditionally, to ensure that the iterate is close enough to $z(\eta)$, one upper bounds the size of the Newton step $n(x) = -H(x)^{-1}g(x)$ in the local norm by a small constant (see Theorem B.2). In order to obtain a quantum speedup, we avoid computing the Hessian exactly, instead we work with a spectral approximation $Q(x) \preceq H(x) \preceq CQ(x)$ for some $C \in \widetilde{O}(1)$. Based on $Q(x)$ one can define $d(x) = Q(x)^{-1}g(x)$ and one of its key properties is the inequality

$$\|n(x)\|_x \leq \|d(x)\|_{Q(x)}.$$

We can thus ensure that the exact Newton step is small by controlling $\|d(x)\|_{Q(x)}$. In Definition 6.6 we give a precise definition of the type of approximate Newton step that we work with (we also allow an approximation of $g(x)$). The purpose of the next section is to analyze Algorithm 7, an algorithm that precisely implements one path-following iteration: starting from an $x \in D$ and η for which the approximate Newton step is small, it updates η and finds a new x that has a small approximate Newton step with respect to the new η.

6.2 Approximate Newton's method

Here we analyze an IPM based on *approximate* Newton steps, which we formalize as follows.

Definition 6.6 (Approximate Newton step)**.** *Let f be a self-concordant barrier with gradient $g(x)$ and Hessian $H(x)$ and suppose there exists a function $Q(x)$ and a real parameter $C \geq 1$ such that*

$$Q(x) \preceq H(x) \preceq CQ(x). \tag{7}$$

Now define

$$d(x) = -Q(x)^{-1}g(x),$$

and let $\tilde{d}(x)$ be a ζ-approximation of $d(x)$ in the local norm, i.e., $\|\tilde{d}(x) - d(x)\|_x \leq \zeta$. We say that $\tilde{d}(x)$ is a (ζ, C)-approximate Newton step *for f at x.*

In our algorithms, ζ and C do not change between iterations and hence we often drop the prefix (ζ, C). Note that we can obtain an approximate Newton step by approximating the gradient in the *inverse-local* norm. Indeed, if $\tilde{g} \in \mathbb{R}^d$ is such that

$$\|\tilde{g} - g(x)\|_{H(x)^{-1}} \leq \zeta, \tag{8}$$

then for $\tilde{d}(x) = -Q(x)^{-1}\tilde{g}(x)$ we have $\|\tilde{d}(x) - d(x)\|_x \leq \zeta$.

The following algorithm implements a single iteration of path-following (going from η to η') using approximate Newton steps. Recall that closeness to the central path can be ensured by having a small (approximate) Newton step. The algorithm applies to a self-concordant function f with

Algorithm 7: The approximate short-step iteration

Parameters: $\alpha, \beta, \delta, \zeta > 0$, such that $(\alpha + \zeta)\beta + (\beta - 1)\sqrt{\vartheta_f} \leq 1/4$
Input: $x \in D$ and $\eta > 0$ with (ζ, C)-approximate Newton step $\tilde{d}_\eta(x)$ satisfying $\|\tilde{d}_\eta(x)\|_{Q(x)} \leq \alpha < 1$
Output: $x' \in D$ with approximate Newton step $\tilde{d}_{\eta'}(x')$ satisfying $\|\tilde{d}_{\eta'}(x')\|_{Q(x')} \leq \alpha$

Set $\eta' = \beta\eta$, $\delta = 1/(4C)$, $x' = x$ and let $\tilde{d}_{\eta'}(x')$ be an approximate Newton step from x';
while $\|\tilde{d}_{\eta'}(x)\|_{Q(x)} \geq \alpha$ **do**
  Let $x' \leftarrow x' + \delta\tilde{d}_{\eta'}(x')$ and let $\tilde{d}_{\eta'}(x')$ be an approximate Newton step from x';
end while
return x'

complexity ϑ_f, we define $d(x)$ as above and use a subscript η to emphasize that the gradient of f_η depends on η (the Hessian of f_η does not depend on η, so neither does $Q(x)$).

The correctness and complexity of the algorithm are established in the following theorem.

Theorem 6.7. *Algorithm 7 correctly returns $x' \in D$ with approximate Newton step $\tilde{d}_{\eta'}(x')$ satisfying $\|\tilde{d}_{\eta'}(x')\|_{Q(x')} \leq \alpha$. The algorithm terminates after $O(C^2)$ iterations, and hence requires $O(C^2)$ many (ζ, C)-approximate Newton steps.*

Combined with the discussion on path-following, this theorem implies Theorem 6.1.

We first sketch the structure of the proof Theorem 6.7 and give two useful lemmas (whose proofs we defer to Appendix B). To analyze the algorithm, we show that after updating η, the while-loop terminates in a roughly constant number of iterations. To do so, we establish two properties: (1) each iteration decreases the function value by at least a certain amount, and (2) at the start of the while-loop, the function value is close to its minimum.

Lemma 6.8. *Assume f is self-concordant, $x \in D$, and $Q(x)$ is such that $Q(x) \preceq H(x) \preceq CQ(x)$ for $C \geq 1$. Let $d(x) = -Q(x)^{-1}g(x)$ and assume $\tilde{d}(x)$ is such that $\|\tilde{d}(x) - d(x)\|_x \leq \zeta$. Let $\delta = 1/(4C)$. If $\delta\|\tilde{d}(x)\|_x \leq 1/2$ and $\zeta \leq \frac{1}{C}\|\tilde{d}(x)\|_x$, then we have*

$$f(x + \delta\tilde{d}(x)) \leq f(x) - \frac{1}{16C^2}\|\tilde{d}(x)\|_x^2.$$

In each iteration that $\|\tilde{d}(x)\|_{Q(x)} > \alpha$, we have $\|\tilde{d}(x)\|_x > \alpha$ and thus Lemma 6.8 shows that

$$f(x + \delta\tilde{d}(x)) \leq f(x) - \frac{\alpha^2}{16C^2}$$

for a suitable choice of δ. It remains to show that initially, after updating η to $\eta' = \beta\eta$, $f_{\eta'}(x)$ is close to the minimum of $f_{\eta'}$. The following lemma is a standard bound on the optimality gap in terms of the size of the Newton step.

Lemma 6.9. *Assume f is self-concordant and let $x \in D$. If $\|n(x)\|_x \leq 1/4$, then f has a minimizer z and*

$$f(x) - f(z) \leq \|n(x)\|_x^2 + 3\left(\frac{\|n(x)\|_x}{1 - \|n(x)\|_x}\right)^3.$$

We are now ready to prove Theorem 6.7.

Proof of Theorem 6.7. From Lemma 6.8 we obtain that in each iteration of the while loop, the function value of f decreases by at least $(\frac{\alpha}{4C})^2$. To upper bound the number of such iterations, Lemma 6.9 shows that it suffices to prove that at the start of the while-loop $\|n_{\eta'}(x)\|_x$ is small. To do so, we first recall the following useful inequalities. If $Q(x) \preceq H(x) \preceq CQ(x)$, then for any $x \in D$ and $v \in \mathbb{R}^d$ we have

$$\|v\|_{Q(x)} \leq \|v\|_x \leq \sqrt{C}\|v\|_{Q(x)} \quad \text{and} \quad \|n(x)\|_x \leq \|d(x)\|_{Q(x)}.$$

Then, as a first step, note that if $\|\tilde{d}_\eta(x)\|_{Q(x)} \leq \alpha$ and $\|\tilde{d}_\eta(x) - d_\eta(x)\|_x \leq \zeta$, then $\|d_\eta(x)\|_{Q(x)} \leq \alpha + \zeta$ and therefore $\|n_\eta(x)\|_x \leq \alpha + \zeta$. The second step is to bound the size of the Newton step after updating η. For this we use the following relation:

$$n_{\eta'}(x) = \frac{\eta'}{\eta} n_\eta(x) + \left(\frac{\eta'}{\eta} - 1\right) g_x(x).$$

Using this relation, the above bound on $n_\eta(x)$, and the triangle inequality, this gives us

$$\|n_{\eta'}(x)\|_x \leq \beta\|n_\eta(x)\|_x + (\beta - 1)\|g_x(x)\|_x \leq (\alpha + \zeta)\beta + (\beta - 1)\sqrt{\vartheta_f} =: \gamma.$$

Recall that by assumption $\gamma \leq 1/4$ and therefore Lemma 6.9 applied to $f_{\eta'}$ shows that

$$f_{\eta'}(x) - \min_{z \in D} f_{\eta'}(z) \leq \gamma^2 + 3\left(\frac{\gamma}{1-\gamma}\right)^3 \leq 7/64.$$

Choosing $\alpha, \beta, \delta, \zeta > 0$ appropriately shows that the while-loop ends after at most $O(C^2)$ iterations. Suitable choices of parameters are for example $\alpha = 1/32$, $\zeta = 1/32$, and $\beta = 1 + \frac{1}{8\sqrt{\vartheta_f}} \leq 2$ so that $\gamma = (\alpha + \zeta)\beta + (\beta - 1)\sqrt{\vartheta_f} \leq 1/4$, and $\delta = \frac{1}{4C}$ so that also $\delta\|\tilde{d}_{\eta'}(x)\|_x \leq \delta(\|d_{\eta'}(x)\|_x + \zeta) \leq \delta(\sqrt{C}\|n_{\eta'}(x)\|_x + \zeta) \leq \delta(\sqrt{C}\gamma + \zeta) < 1/2$. □

Remark 6.10 (Maintaining feasibility)**.** *A key property of self-concordant barriers is that at each interior point, the unit ball in the local norm is contained in the domain. This ensures that we maintain feasibility of the iterates since our steps $\delta\tilde{d}(x)$ satisfy $\delta\|\tilde{d}(x)\|_x < 1$ (indeed, $\delta < 1$ and the approximate Newton step has size $< 1/2$, even after updating η). A second crucial assumption that we have made is that the domain is full-dimensional; this allows us to avoid (potentially costly) projections onto subspaces.*

7 Quantumly approximating Hessians and gradients

From Theorem 6.1 it follows that we can implement an IPM by implementing approximate Newton steps, which can be derived from approximations of the gradient and Hessian of the barrier function. Specifically, from a feasible point $x \in \mathbb{R}^d$ we need to compute approximations $Q(x)$ and $g(x)$ satisfying

$$Q(x) \preceq H(x) \preceq CQ(x) \quad \text{and} \quad \|\tilde{g}(x) - g(x)\|_{H(x)^{-1}} \leq \zeta.$$

for parameters C and ζ satisfying $C, \zeta^{-1} \in \widetilde{O}(1)$. We first describe the high-level strategy for computing $Q(x)$ and $\tilde{g}$ and then in the next three subsections we give a detailed analysis of the complexity for the logarithmic, volumetric and Lewis weight barriers.

At a high level, for each of the barriers we will use our quantum spectral approximation algorithm (Theorem 3.1) to obtain a $\widetilde{O}(1)$-spectral approximation $Q(x)$ of the Hessian. This theorem shows how to obtain a spectral approximation of $B^T B$ using row-queries to B. Our Hessians all take the form $A^T S_x^{-1} W S_x^{-1} A$ where S_x is the diagonal matrix containing the slacks and $W \succ 0$ is a diagonal weight matrix. For the log-barrier, the volumetric barrier and the Lewis-weight barrier, the matrix W is respectively the identity, the diagonal matrix of leverage scores of $S_x^{-1} A$, and the diagonal matrix of Lewis weights of $S_x^{-1} A$. For the latter two barriers we thus need to compute constant factor approximations of leverage scores and Lewis weights respectively. We do so using Theorem 3.2 and Theorem 4.5 respectively.

To obtain a good estimate of the gradient, we use our quantum algorithm for approximate matrix-vector products (Theorem 5.1). To apply the theorem we note that $g(x)$ takes the form $-A^T S_x^{-1} W \mathbf{1}$, with $\mathbf{1}$ the all-ones vector and W the same rescaling matrix as before. We interpret this as a matrix-vector product between $B = \sqrt{W} S_x^{-1} A$ and $v = \sqrt{W}\mathbf{1}$. This means that for the volumetric barrier and the Lewis weight barrier we cannot afford to query entries of v exactly, instead we will query multiplicative approximations of these entries. The next lemma analyzes the error that such a multiplicative approximation incurs.

Lemma 7.1. *Let $B \in \mathbb{R}^{n \times d}$, $D \in \mathbb{R}^{n \times n}$ diagonal with $D_{ii} \in [1-\varepsilon, 1+\varepsilon]$, and $v \in \mathbb{R}^n$, then*

$$\|B^T v - B^T D v\|_{(B^T B)^{-1}} \leq \varepsilon \|v\|_2$$

Proof. We bound the squared norm as follows:

$$\begin{aligned} \|B^T v - B^T D v\|^2_{(B^T B)^{-1}} &= \|B^T (D - I) v\|^2_{(B^T B)^{-1}} \\ &= v^T (D - I) B (B^T B)^{-1} B^T (D - I) v \\ &\leq v^T (D - I)^2 v \leq \varepsilon^2 \|v\|_2^2 \end{aligned}$$

where in the first inequality we use that $B(B^T B)^{-1} B^T$ is a projector. □

For both the volumetric and Lewis weight barrier we have $\|v\|_2 \leq \sqrt{d}$. To achieve a small constant error in the gradient, it thus suffices to use $(1 \pm O(1/\sqrt{d}))$-multiplicative approximations of the entries of v.

7.1 Logarithmic barrier

The logarithmic barrier (cf. [41]) for a polytope $Ax \geq b$, with $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, is defined as

$$F(x) := -\sum_{i=1}^{n} \log(a_i^T x - b_i),$$

which is well defined in the interior of the polytope. It is a self-concordant barrier with complexity [42, Section 2.3.1]

$$\vartheta_F \in O(n).$$

Its gradient and Hessian are

$$g_F(x) \coloneqq \nabla F(x) = -A^T S_x^{-1} \mathbf{1} \quad \text{and} \quad H_F(x) \coloneqq \nabla^2 F(x) = A^T S_x^{-2} A.$$

The following lemmas bound the cost of approximating these using our quantum algorithms.

Lemma 7.2 (Hessian of logarithmic barrier)**.** *Given query access to $A \in \mathbb{R}^{n\times d}$, $x \in \mathbb{R}^d$ and $b \in \mathbb{R}^n$, and a parameter $\varepsilon > 0$, we give a quantum algorithm that with high probability computes a matrix $Q(x)$ such that $Q(x) \approx_\varepsilon H_F(x)$. The algorithm uses $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to A, x, and b and time $\widetilde{O}(\sqrt{nd}r/\varepsilon + d^\omega)$.*

Proof. We apply Theorem 3.1 to the matrix $S_x^{-1}A$ and observe that we can implement one row-query to $S_x^{-1}A$ in time $O(r)$. Indeed, it suffices to observe that we can use $O(1)$ row queries and $O(r)$ time to compute the slack of a single constraint. □

Lemma 7.3 (Gradient of logarithmic barrier)**.** *Given query access to $A \in \mathbb{R}^{n\times d}$, $x \in \mathbb{R}^d$ and $b \in \mathbb{R}^n$, we give a quantum algorithm that with high probability computes a vector $\tilde{g}(x)$ such that $\|\tilde{g}(x) - g_F(x)\|_{H_F(x)^{-1}} \le \zeta$ using $\widetilde{O}(\sqrt{n}d/\zeta)$ row queries to A, x, and b and time $\widetilde{O}(\sqrt{n}d^2r/\zeta + d^\omega)$.*

Proof. As in the proof of Lemma 7.2, we set $B = S_x^{-1}A$ and observe that we can implement one row query to $S_x^{-1}A$ in time $O(r)$. We then use Theorem 5.1 with this B and $v = \mathbf{1}$ to obtain a vector $\tilde{y}$ such that $\|\tilde{y} - g_F(x)\|_{H_F(x)^{-1}} = \|\tilde{y} - g_F(x)\|_{(B^TB)^{-1}} \le \zeta$. □

By Theorem 6.1 we can solve an LP, returning a feasible point ε-close to optimal, while making $O(\sqrt{\vartheta_f}\log(1/\varepsilon))$ many approximate gradient and Hessian queries. Combining Lemma 7.2 and Lemma 7.3 (with $\varepsilon, \zeta = 1/\operatorname{polylog}(n)$), and using the fact that $\vartheta_F \in O(n)$, this yields a quantum IPM based on the logarithmic barrier which makes $\widetilde{O}(nd)$ row queries and uses time $\widetilde{O}(nd^2r + \sqrt{n}d^\omega)$. While useful as a proof of concept, the query complexity is trivial, and the time complexity is worse than the best classical algorithm which runs in time $\widetilde{O}(nd + d^3)$ [11]. We hence turn to a more complicated barrier.

7.2 Volumetric barrier

The volumetric barrier for a polytope $Ax \ge b$, introduced by Vaidya [47] (see also [48, 1]), is defined as

$$V(x) \coloneqq \frac{1}{2}\ln\det(H_F(x)) = \frac{1}{2}\ln\det(A^T S_x^{-2} A).$$

The volumetric barrier is a self-concordant barrier with complexity

$$\vartheta_V \in O(\sqrt{nd}).$$

An appropriately weighted linear combination of the volumetric barrier and the logarithmic barrier, called the *hybrid barrier*, has complexity parameter $O(\sqrt{nd})$. For ease of notation, we focus here on approximating the Hessian and gradient of the volumetric barrier.[12]

[12]The hybrid barrier is $V_\rho(x) = V(x) + \rho F(x)$ where $\rho = (d-1)/(n-1)$, following the notation of [1]. The analogue of Eq. (9) thus holds for the matrix $\widetilde{H}_{V_\rho}(x) = A^T S_x^{-1}(\Sigma_x + \rho I)S_x^{-1}A$, and the gradient is $g_{V_\rho}(x) = A^T S_x^{-1}(\Sigma_x + \rho I)\mathbf{1}$. To obtain Lemmas 7.4 and 7.5 for the hybrid barrier it thus suffices to replace Σ_x by $\Sigma_x + \rho I$ in the proofs (and use the fact that $a \approx_\varepsilon b$ implies $a + \rho \approx_\varepsilon b + \rho$ for $a, b, \rho > 0$).

Given a feasible $x \in \mathbb{R}^d$, let σ_x be the vector of leverage scores of $S_x^{-1}A$ and let $\Sigma_x = \mathrm{Diag}(\sigma_x)$. Then the gradient of V at x has the simple form

$$g_V(x) \coloneqq \nabla V(x) = -A^T S_x^{-1} \Sigma_x \mathbf{1}.$$

The Hessian of V has a more complicated form, but it admits a constant factor spectral approximation that has a simple form. We have

$$\widetilde{H}_V(x) \preceq H_V(x) \preceq 5\widetilde{H}_V(x), \tag{9}$$

where

$$\widetilde{H}_V(x) = A^T S_x^{-1} \Sigma_x S_x^{-1} A.$$

We can efficiently approximate this approximate Hessian.

Lemma 7.4 (Hessian of volumetric barrier)**.** *Given query access to $A \in \mathbb{R}^{n\times d}$, $x \in \mathbb{R}^d$ and $b \in \mathbb{R}^n$, we give a quantum algorithm that computes with high probability a matrix $Q(x)$ such that $Q(x) \approx_\varepsilon \widetilde{H}_V(x)$. The algorithm uses $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to A, x, and b and time $\widetilde{O}(\sqrt{nd}r^2/\varepsilon + d^\omega)$.*

Proof. We use Theorem 3.2 to set up query access to $\tilde{\sigma}_x$, a vector of $(1 \pm \varepsilon/3)$-multiplicative approximations to the leverage scores of $S_x^{-1}A$. This requires a preprocessing that uses $\widetilde{O}(\sqrt{nd}/\varepsilon)$ row queries to $S_x^{-1}A$ and time[13] $\widetilde{O}(\sqrt{nd}r/\varepsilon + d^\omega/\varepsilon^2)$; afterwards we can query an entry of $\tilde{\sigma}_x$ using one row query to $S_x^{-1}A$ and time $\widetilde{O}(r^2)$. Let $\widetilde{\Sigma}_x = \mathrm{Diag}(\tilde{\sigma}_x)$.

We then apply Theorem 3.1, with approximation factor $\varepsilon/3$, to the matrix $\widetilde{\Sigma}_x^{1/2} S_x^{-1} A$ and observe that we can implement one row query to $\widetilde{\Sigma}_x^{1/2} S_x^{-1} A$ using one query to $\widetilde{\sigma}_x$ and one row query to $S_x^{-1}A$. The cost of row queries to the latter follows from the observation that we can use $O(1)$ row queries to A, b, and time $O(r)$, to compute the slack of a single constraint. Let B be the matrix returned by the algorithm from Theorem 3.1. With high probability, B has $\widetilde{O}(d)$ rows and satisfies

$$B^T B \approx_{\varepsilon/3} A^T S_x^{-1} \widetilde{\Sigma}_x S_x^{-1} A.$$

Since we also have

$$A^T S_x^{-1} \widetilde{\Sigma}_x S_x^{-1} A \approx_{\varepsilon/3} \widetilde{H}_V(x),$$

this shows that

$$B^T B \approx_\varepsilon \widetilde{H}_V(x),$$

where we used that $1 - \varepsilon \le (1 - \varepsilon/3)^2$ and $(1 + \varepsilon/3)^2 \le 1 + \varepsilon$. □

Lemma 7.5 (Gradient of volumetric barrier)**.** *Given query access to $A \in \mathbb{R}^{n\times d}$, $x \in \mathbb{R}^d$ and $b \in \mathbb{R}^n$, we give a quantum algorithm that with high probability computes a vector $\tilde{g}(x)$ such that $\|\tilde{g}(x) - g(x)\|_{H_V(x)^{-1}} \le \zeta$ using $\widetilde{O}(\sqrt{n}d/\zeta)$ row queries to A, x and b, and time $\widetilde{O}(\sqrt{n}d^2r/\zeta + d^{\omega+1}/\zeta^2)$.*

Proof. Let $\delta > 0$ be a parameter that we choose later. We first use Theorem 3.2 to set up query access to $\tilde{\sigma}_x$, a vector of $(1 \pm \delta)$-multiplicative approximations to the leverage scores of $S_x^{-1}A$. We

[13] For convenience we replace the term $\min\{d^\omega, dr^2/\varepsilon^2\}$ by d^ω.

then let $\widetilde{B} = \widetilde{\Sigma}_x^{1/2} S_x^{-1} A$ and $\tilde{v} = \widetilde{\Sigma}_x^{1/2} \mathbf{1}$ and use Theorem 5.1 to obtain a vector $\tilde{y} \in \mathbb{R}^d$ such that $\|\tilde{y} - \widetilde{B}^T \tilde{v}\|_{(\widetilde{B}^T \widetilde{B})^{-1}} \leq \zeta$. By Eq. (9) and a triangle inequality we can bound

$$\|\tilde{y} - g(x)\|_{H_V(x)^{-1}} \leq \|\tilde{y} - g(x)\|_{(\widetilde{B}^T \widetilde{B})^{-1}} \leq \|\tilde{y} - \widetilde{B}^T \tilde{v}\|_{(\widetilde{B}^T \widetilde{B})^{-1}} + \|g(x) - \widetilde{B}^T \tilde{v}\|_{(\widetilde{B}^T \widetilde{B})^{-1}}.$$

For the first term, we have the upper bound $\|\tilde{y} - \widetilde{B}^T \tilde{v}\|_{(\widetilde{B}^T \widetilde{B})^{-1}} \leq \zeta$ by construction. For the second term, we first rewrite $g(x)$ as

$$g(x) = A^T S_x^{-1} \Sigma_x \mathbf{1} = (A^T S_x^{-1} \widetilde{\Sigma}_x^{1/2})(\widetilde{\Sigma}_x^{-1/2} \Sigma_x \mathbf{1}) = \widetilde{B}^T v'$$

where $v' = \widetilde{\Sigma}_x^{-1/2} \Sigma_x \mathbf{1}$. We then note that $v' = \Sigma_x \widetilde{\Sigma}_x^{-1} \tilde{v}$, i.e., entrywise v' is a $(1 \pm \delta)$-approximation of $\tilde{v}$. Lemma 7.1 thus shows that $\|g(x) - \widetilde{B}^T \tilde{v}\|_{(\widetilde{B}^T \widetilde{B})^{-1}} \leq \delta \|\tilde{v}\|_2 \leq \delta \sqrt{(1+\delta)d}$, where the last inequality uses that $\|\tilde{v}\|_2 = \sqrt{\sum_{i \in [n]} (\tilde{\sigma}_x)_i} \leq \sqrt{(1+\delta)d}$. Setting $\delta = \zeta / \sqrt{d}$ shows that $\|\tilde{g} - g(x)\|_{H_V(x)^{-1}} = O(\zeta)$.

We finally establish the query and time complexity. We use Theorem 3.2 to set up query access to $\tilde{\sigma}_x$. Since $1/\delta^2 > r$, it is advantageous to use the first (direct) approach that uses as preprocessing $\widetilde{O}(\sqrt{n} d/\zeta)$ row queries to $S_x^{-1} A$ and time $\widetilde{O}(d^{\omega+1}/\zeta^2 + r\sqrt{n} d/\zeta)$, and then allows us to query an entry of $\tilde{\sigma}_x$ at a cost of 1 row query to $S_x^{-1} A$ and time $O(r^2)$. The application of Theorem 5.1 uses $\widetilde{O}(\sqrt{n} d/\zeta)$ row queries to $\widetilde{\Sigma}_x^{1/2} S_x^{-1} A$ and time $\widetilde{O}(r\sqrt{n} d^2/\zeta + d^\omega)$ (here we are using that the time needed per query to $\tilde{\sigma}_x$ is r^2 and thus less than dr). □

By Theorem 6.1 we can solve an LP to precision ε while making $O(\sqrt{\vartheta_f} \log(1/\varepsilon))$ many approximate gradient and Hessian queries. Combining Lemma 7.4 and Lemma 7.5 (with $\varepsilon, \zeta = 1/\operatorname{polylog}(n)$), and using the fact that $\vartheta_V \in O(\sqrt{nd})$, this yields a quantum IPM based on the volumetric barrier which makes $\widetilde{O}(n^{3/4} d^{5/4})$ row queries and uses time

$$\widetilde{O}(n^{1/4} d^{1/4} (\sqrt{n} d^2 r + d^{\omega+1})),$$

which is $n^{3/4} \operatorname{poly}(d, \log(n), 1/\varepsilon)$.[14] For n sufficiently large the query and time complexity are sublinear and beat that of classical algorithms. However, the $n^{3/4}$-scaling is worse than the $\sqrt{n}$-scaling of quantum algorithms based on the cutting plane method (Appendix A). Hence we turn to the final barrier.

7.3 Lewis weight barrier

Finally, we discuss the Lewis weight barrier, as introduced by Lee and Sidford in [31]. For integer $p > 0$, it is defined as

$$\psi(x) = \ln \det(A^T S_x^{-1} W_x^{1 - \frac{2}{p}} S_x^{-1} A) \quad \text{where } W_x = \operatorname{Diag}(w^{(p)}(S_x^{-1} A)).$$

Its gradient is

$$g_\psi(x) := \nabla \psi(x) = -A^T S_x^{-1} W_x \mathbf{1}.$$

While the exact Hessian might again take a complicated form, we can approximate it by

$$\widetilde{H}_\psi(x) := A^T S_x^{-1} W_x S_x^{-1} A.$$

[14] Strictly speaking, this complexity uses the hybrid barrier, see Footnote 12.

Indeed, it holds that

$$\widetilde{H}_\psi(x) \preceq \nabla^2\psi(x) \preceq (1+p)\widetilde{H}_\psi(x).$$

Letting $v_p = (p+2)^{3/2} n^{\frac{1}{p+2}} + 4\max\{p,2\}^{2.5}$, Lee and Sidford showed [31, Theorem 30] that the barrier function $v_p^2 \cdot \psi(x)$ is a self-concordant barrier function with complexity $\vartheta_\psi \in O(dv_p^2)$. We will be considering $p = \mathrm{polylog}(n)$, in which case the barrier has complexity

$$\vartheta_\psi \in \widetilde{O}(d).$$

For the complexity statements that follow, we assume $p = \mathrm{polylog}(n)$ so that polynomial factors in p are absorbed in the $\widetilde{O}$-notation.

To approximate the Hessian and gradient for the Lewis weight barrier, we follow the same approach as for the volumetric barrier, but with leverage scores replaced by Lewis weights. The only part of the proofs that changes is the complexity analysis.

Lemma 7.6 (Hessian of the Lewis weight barrier)**.** *Given query access to $A \in \mathbb{R}^{n\times d}$, $x \in \mathbb{R}^d$ and $b \in \mathbb{R}^n$, we give a quantum algorithm that with high probability computes a matrix $Q(x)$ such that $Q(x) \approx_\varepsilon \widetilde{H}_\psi(x)$. The algorithm uses $\widetilde{O}(\sqrt{n}d^{3/2}/\varepsilon^2)$ row queries to A, x, and b and time $\widetilde{O}\left(\frac{d^{3/2}}{\varepsilon^3}\left(r^2\sqrt{nd} + d^\omega\right)\right)$.*

Proof. We follow the same proof strategy as the one we used for Lemma 7.4. We refer to that proof for correctness of the choices of parameters that follow. We use Theorem 4.5 to set up query access to $\tilde{w}_x$, a vector of $(1\pm\varepsilon/3)$-multiplicative approximations to the ℓ_p-Lewis weights of $S_x^{-1}A$. This requires a preprocessing of $\widetilde{O}(\sqrt{n}d^{3/2}/\varepsilon^2)$ row queries to $S_x^{-1}A$ and time $\widetilde{O}(\frac{d^{3/2}}{\varepsilon^3}\cdot\left(r^2\sqrt{nd}+d^\omega\right))$. Each query to $\tilde{w}_x$ then requires 1 row-query to $S_x^{-1}A$ and time $\widetilde{O}(r^2d^{1/2}/\varepsilon)$. We then apply Theorem 3.1, with desired approximation factor $\varepsilon/3$, to the matrix $\widetilde{W}_x^{1/2}S_x^{-1}A$ and observe that we can implement one row query to $\widetilde{W}_x^{1/2}S_x^{-1}A$ using one query to $\tilde{w}_x$ and one row query to $S_x^{-1}A$. The cost of row queries to the latter follows from the observation that we can use $O(1)$ row queries and time $O(r)$ to compute the slack of a single constraint. Observe that the query and time complexity of setting up query access to $\tilde{w}_x$ dominate the complexity of the algorithm. □

Lemma 7.7 (Gradient of the Lewis weight barrier)**.** *Given query access to $A \in \mathbb{R}^{n\times d}$, $x \in \mathbb{R}^d$ and $b \in \mathbb{R}^n$, we give a quantum algorithm that with high probability computes a vector $\tilde{g}(x) \in \mathbb{R}^d$ such that $\|\tilde{g}(x) - g(x)\|_{H(x)^{-1}} \le \zeta$ using $\widetilde{O}(\sqrt{n}d^{5/2}/\zeta^2)$ row queries to A, x, and b and time $\widetilde{O}\left(\left(\sqrt{n}d^{7/2}r^2 + d^{\omega+3}\right)/\zeta^3\right)$.*

Proof. We follow the same proof strategy as the one we used for Lemma 7.5. We refer to that proof for correctness of the choices of parameters that follow. The dominant term in both the query and time complexity is the use of Theorem 4.5 to set up query access to $\tilde{w}_x$, a vector of $(1\pm\zeta/\sqrt{d})$-multiplicative approximations to the leverage scores of $S_x^{-1}A$. This requires a preprocessing of $\widetilde{O}(\sqrt{n}d^{5/2}/\zeta^2)$ row queries to $S_x^{-1}A$ and time[15] $\widetilde{O}(\sqrt{n}d^{7/2}r^2/\zeta^3 + d^{\omega+3}/\zeta^3)$. Each query to $\tilde{w}_x$ then requires 1 row-query to $S_x^{-1}A$ and time $\widetilde{O}(r^2d/\zeta)$. As before, we then apply Theorem 5.1 to $\widetilde{W}_x^{1/2}S_x^{-1}A$ and $\widetilde{W}_x^{1/2}\mathbf{1}$ with desired precision ζ. This uses $\widetilde{O}(\sqrt{n}d/\zeta)$ row queries to $\widetilde{W}_x^{1/2}S_x^{-1}A$ and time $\widetilde{O}((r^2d/\zeta)\sqrt{n}d/\zeta + d^\omega)$. Note that unlike in the proof of Lemma 7.5, here the time needed per query to W_x is r^2d/ζ and this exceeds the additional computational cost of dr that the multivariate

[15] For convenience we replace the term $\min\{d^{\omega+3/2}, d^{5/2}r^2\}$ by $d^{\omega+3/2}$.

quantum mean estimation algorithm uses per quantum sample. However, as for the previous lemma, observe that the query and time complexity of setting up query access to $\tilde{w}_x$ dominates the complexity of the algorithm. □

Again invoking Theorem 6.1, we can solve an LP to precision ε while making $O(\sqrt{\vartheta_f}\log(1/\varepsilon))$ many approximate gradient and Hessian queries. Combining Lemma 7.6 and Lemma 7.7 (with $\varepsilon, \zeta = 1/\operatorname{polylog}(n)$), and using the fact that $\vartheta_V \in \widetilde{O}(d)$, this yields a quantum IPM based on the volumetric barrier which makes $\widetilde{O}(\sqrt{n}d^3)$ row queries and uses time

$$\widetilde{O}(\sqrt{d}(\sqrt{n}d^{7/2}r^2 + d^{\omega+3})),$$

which is $\sqrt{n}\operatorname{poly}(d, \log(n), 1/\varepsilon)$. This proves our main Theorem 1.1. For n sufficiently large the query and time complexity are sublinear and beat that of classical algorithms. The $\sqrt{n}$-scaling matches that of a quantum algorithm based on the cutting plane method (Appendix A), although the d-dependence is significantly worse. See Section 1.3 for a discussion on this benchmark comparison, and for open directions on how to improve the d-dependency of our IPM.

8 Lower bounds

Here we discuss some relevant lower bounds. In Section 8.1 we show that the complexity of our algorithm for spectrally approximating A^TA is tight in terms of the number of row queries to A: $\Omega(\sqrt{nd})$ quantum row queries are needed. In Section 8.2 we reproduce a bound from [3, Cor. 30] which shows that $\Omega(\sqrt{ndr})$ queries to A are needed to solve LPs up to constant additive error, even when the entries of A, b, and c are restricted to $\{-1, 0, 1\}$.

8.1 Spectral approximation lower bound

Here we discuss two complementary lower bounds for spectral approximation. The first one applies to arbitrary $n \geq d$ but constant ε. It easily proven by a reduction to quantum search.

Theorem 8.1. *For arbitrary $n \geq d$, there is a family of matrices $A \in \mathbb{R}^{n\times d}$ for which finding a constant factor spectral approximation of A^TA requires $\Omega(\sqrt{nd})$ row queries to A.*

Proof. Without loss of generality, assume d divides n. Given d bitstrings $z_1, \ldots, z_d \in \{0,1\}^{n/d}$ we construct a matrix A that consists of d blocks of size $n/d \times d$: the jth block contains z_j in the jth column and is zero elsewhere. Note that a row query to A can be simulated with one query to one of the z_j's. Moreover, the matrix A^TA has a particularly simple form: $A^TA = \operatorname{Diag}(|z_1|, |z_2|, \ldots, |z_d|)$. The diagonal of a spectral $1/2$-approximation of A^TA contains $1/2$-approximations of the diagonal of A^TA and therefore allows us to compute the string $(\mathrm{OR}(z_1), \mathrm{OR}(z_2), \ldots, \mathrm{OR}(z_d))$. To conclude the proof, we observe that computing the string $(\mathrm{OR}(z_1), \mathrm{OR}(z_2), \ldots, \mathrm{OR}(z_d))$ is known[16] to require $\Omega(d\sqrt{n/d}) = \Omega(\sqrt{nd})$ quantum queries to the bitstrings $z_1, \ldots, z_d$. □

The second lower bound also shows optimality of the ε-dependency, but it is restricted to $n \in O(d^2)$. The bound follows from a lower bound on graph sparsification in [5]. There it is shown that

[16] This follows for instance from combining the $\Omega(\sqrt{n/d})$ lower bound on OR over n/d bits [9] with the strong direct product theorem for quantum query complexity [29].

any quantum algorithm for computing an ε-spectral sparsifier of a graph G must make $\Omega(\sqrt{nd}/\varepsilon)$ queries to its adjacency list, where d is the number of vertices of G, $n \in O(d^2)$ is the number of edges of G, and $\varepsilon \in \Omega(\sqrt{d/n})$. This lower bound applies to our setting if we let A denote the edge-vertex incidence matrix of a graph G, in which case an adjacency list query can be reduced to a row query to A, and returning an ε-spectral approximation of A is equivalent to returning an ε-spectral sparsifier of G. We hence get the following lower bound.

Lemma 8.2 ([5, Theorem 2])**.** *For arbitrary d, $n \in O(d^2)$ and $\varepsilon \in \Omega(\sqrt{d/n})$, there is a family of matrices $A \in \mathbb{R}^{n\times d}$ for which finding an ε-spectral approximation of A^TA requires $\widetilde{\Omega}(\sqrt{nd}/\varepsilon)$ row queries to A.*

8.2 LP solving lower bound

The following is a small variation of [3, Cor. 30] where we include the row sparsity r as a parameter. Their lower bound corresponds to setting $r = d$. Since the bound below is only a minor variation of their bound, we only give a sketch of how to recover it from their arguments (one mainly has to take care to transform their LP from canonical form to the form we use).

Theorem 8.3 (Variation of [3, Cor. 30])**.** *There is a family of linear programs of the form* $\min c^Tx$ *s.t.* $Ax \geq b$ *with $A \in \mathbb{R}^{n\times d}$ for which determining the optimal value up to additive error $< 1/2$, with probability $\geq 2/3$, requires at least $\Omega(\sqrt{ndr})$ quantum queries to A.*

Proof sketch. In [3] the authors obtain a quantum query lower bound on the cost of solving linear programs by constructing a linear program whose optimal value computes a Boolean function.

Concretely, consider the MAJ_a-OR_b-MAJ_c function that takes an input $Z \in \{0,1\}^{a\times b\times c}$ and outputs

$$\begin{aligned}
&\mathrm{MAJ}_a(\\
&\quad \mathrm{OR}_b(\mathrm{MAJ}_c(Z_{111},\dots,Z_{11c}),\dots,\mathrm{MAJ}_c(Z_{1b1},\dots,Z_{1bc})),\\
&\quad \dots,\\
&\quad \mathrm{OR}_b(\mathrm{MAJ}_c(Z_{a11},\dots,Z_{a1c}),\dots,\mathrm{MAJ}_c(Z_{ab1},\dots,Z_{abc}))\\
&\quad)
\end{aligned}$$

The quantum query complexity of MAJ_a-OR_b-MAJ_c is known to be $\Theta(a\sqrt{bc})$.

We then consider the following linear program where we let Z^i be the $c \times b$ matrix whose jk-th entry is Z_{ikj}:

$$\begin{aligned}
\max\ & \sum_{k=1}^{c} w_k \\
\text{s.t.}\ & \begin{pmatrix} I_c & -Z^1 & -Z^2 & \cdots & -Z^a \\ & \mathbf{1}_b^T & & & \\ & & \mathbf{1}_b^T & & \\ & & & \ddots & \\ & & \mathbf{1}_b^T & & \mathbf{1}_b^T \end{pmatrix} \begin{pmatrix} w \\ v^1 \\ \vdots \\ v^a \end{pmatrix} \leq \begin{pmatrix} 0_c \\ \mathbf{1}_a \end{pmatrix} \\
& w \in \mathbb{R}_+^c, v^1,\dots,v^a \in \mathbb{R}_+^b
\end{aligned}$$

In [3] it is shown that the optimal value of this linear program is $\sum_{i=1}^{a} \max_{j\in[b]} \sum_{\ell=1}^{c} Z_{ij\ell}$; when restricted to the "hard instances" for each of the Boolean functions, this integer determines the value of the MAJ_a-OR_b-MAJ_c function on Z (we refer to [3] for a more detailed description of the conditions on Z).

The number of linear inequality constraints is $c + a$ and the number of nonnegative variables is $c + ab$. Each column of the constraint matrix has at most $c + 1$ nonzero entries. The primal problem thus has more variables than constraints, we therefore consider the dual linear program, which by strong duality has the same objective value. The dual of the above LP will thus have $d = c + a$ nonnegative variables, $c + ab$ constraints, and a constraint matrix that has sparsity $r = c + 1$. To bring the LP into our desired form we view the d nonnegativity constraints as inequality constraints on d real-valued variables. This brings the total number of linear inequality constraints to $n = 2c + a(b + 1)$. We express a, b, c in terms of r, n, d as follows: $c = r - 1$, $a = d - r + 1$ and $b = \frac{n-2(r-1)}{d-r+1} - 1$. Note that as long as $r \leq d/2$, we have $a = \Theta(d)$, $b = \Theta(n/d)$, and $c = \Theta(r)$. This implies that finding an additive $\varepsilon = 1/3$ approximation of the optimal value of the dual LP requires $\Omega(d\sqrt{n/dr}) = \Omega(\sqrt{ndr})$ queries to the constraint matrix A. □

Currently there is a gap between the quantum query lower bound and upper bound. While we expect to be able to improve the upper bound, it is possible that the lower bound can be strengthened. For completeness however, we remark that the same construction as above gives a classical query lower bound that is a factor $\sqrt{b} = \sqrt{n/d}$ larger (since the classical query complexity of OR_b is $\Theta(b)$ instead of the quantum query complexity $\Theta(\sqrt{b})$. In other words, classically the lower bound is $\Omega(nr)$, linear in the input size.

Acknowledgements

Part of the exposition of the quantum algorithm for spectral approximation is based on the report of an internship at IRIF by Hugo Abreu and Hugo Thomas, which we are thankful for. We also thank Adrian Vladu for useful discussions on IPMs and Lewis weights, and Arjan Cornelissen for useful discussions on quantum mean estimation. Simon Apers was partially supported by French projects EPIQ (ANR-22-PETQ-0007), QUDATA (ANR-18-CE47-0010) and QUOPS (ANR-22-CE47-0003-01), and EU project QOPT (QuantERA ERA-NET Cofund 2022-25). Sander Gribling was partially supported by the projects QUDATA (ANR-18-CE47-0010) and QOPT (QuantERA ERA-NET Cofund 2022-25).

References

[1] Kurt M. Anstreicher. Volumetric path following algorithms for linear programming. *Mathematical Programming*, 76:245–263, 1997.

[2] Joran van Apeldoorn and András Gilyén. Improvements in Quantum SDP-Solving with Applications. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 99:1–99:15, 2019.

[3] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum SDP-solvers: Better upper and lower bounds. *Quantum*, 4(230), 2020. Earlier version in FOCS'17. arXiv:1705.01843.

[4] Joran van Apeldoorn and András Gilyén. Quantum algorithms for zero-sum games, 2019. arXiv:1904.03180.

[5] Simon Apers and Ronald de Wolf. Quantum speedup for graph sparsification, cut approximation, and Laplacian solving. *SIAM Journal on Computing*, 51(6):1703–1742, 2022.

[6] Brandon Augustino. *Quantum Algorithms for Symmetric Cones*. PhD thesis, Lehigh University, 2023.

[7] Brandon Augustino, Giacomo Nannicini, Tamás Terlaky, and Luis F Zuluaga. Quantum interior point methods for semidefinite optimization. *Quantum*, 7:1110, 2023.

[8] Adam Bouland, Yosheb M Getachew, Yujia Jin, Aaron Sidford, and Kevin Tian. Quantum speedups for zero-sum games via improved dynamic Gibbs sampling. In *International Conference on Machine Learning*, pages 2932–2952. PMLR, 2023.

[9] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik: Progress of Physics*, 46(4-5):493–505, 1998.

[10] F. L. Brandao and K. M. Svore. Quantum speed-ups for solving semidefinite programs. In *IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 415–426, 2017.

[11] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 775–788, 2020.

[12] 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. In *Proceedings of the 46th International Colloquium on Automata, Languages and Programming (ICALP 2019)*, volume 132, pages 27:1–27:14, 2019.

[13] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132, pages 33:1–33:14, 2019. arXiv:1804.01973.

[14] Kenneth L Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM (JACM)*, 42(2):488–499, 1995.

[15] Kenneth L. Clarkson and David P. Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6), jan 2017.

[16] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190, 2015.

[17] Michael B Cohen and Richard Peng. ℓ_p row sampling by Lewis weights. In *Proceedings of the forty-seventh annual ACM Symposium on Theory of computing*, pages 183–192, 2015.

[18] Arjan Cornelissen, Yassine Hamoudi, and Sofiene Jerbi. Near-optimal quantum algorithms for multivariate mean estimation. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 33–43, 2022.

[19] Arjan Cornelissen, Yassine Hamoudi, and Sofiene Jerbi. Near-optimal quantum algorithms for multivariate mean estimation. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 33–43, New York, NY, USA, 2022. Association for Computing Machinery.

[20] Alexander M Dalzell, B David Clader, Grant Salton, Mario Berta, Cedric Yen-Yu Lin, David A Bader, Nikitas Stamatopoulos, Martin JA Schuetz, Fernando GSL Brandão, Helmut G Katzgraber, and William J Zeng. End-to-end resource analysis for quantum interior point methods and portfolio optimization. *arXiv preprint arXiv:2211.12489*, 2022.

[21] Maryam Fazel, Yin Tat Lee, Swati Padmanabhan, and Aaron Sidford. Computing lewis weights to high precision. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2723–2742, 2022.

[22] Baihe Huang, Shunhua Jiang, Zhao Song, Runzhou Tao, and Ruizhe Zhang. A faster quantum algorithm for semidefinite programming via robust ipm framework. *arXiv preprint arXiv:2207.11154*, 2022.

[23] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[24] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[25] Iordanis Kerenidis and Anupam Prakash. A quantum interior point method for LPs and SDPs. *ACM Transactions on Quantum Computing*, 1(1):1–32, 2020.

[26] Iordanis Kerenidis, Anupam Prakash, and Dániel Szilágyi. Quantum algorithms for second-order cone programming and support vector machines. *Quantum*, 5:427, 2021.

[27] L. G. Khachiyan. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 244:1093–1096, 1979.

[28] Kasper Green Larsen and Jelani Nelson. Optimality of the johnson-lindenstrauss lemma. In *IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 633–638, 2017.

[29] Troy Lee and Jérémie Roland. A strong direct product theorem for quantum query complexity. *computational complexity*, 22:429–462, 2013.

[30] Yin Tat Lee. *Faster Algorithms for Convex and Combinatorial Optimization*. PhD thesis, Massachusetts Institute of Technology, 2016.

[31] Yin Tat Lee and Aaron Sidford. Solving linear programs with sqrt(rank) linear system solves, 2019. arXiv:1910.08033.

[32] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *IEEE 56th Annual Symposium*

on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015, pages 1049–1065, 2015.

[33] D. Lewis. Finite dimensional subspaces of L_p. *Studia Mathematica*, 63(2):207–212, 1978.

[34] Mu Li, Gary L Miller, and Richard Peng. Iterative row sampling. In *IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 127–136, 2013.

[35] Yang Liu and Shengyu Zhang. Fast quantum algorithms for least squares regression and statistic leverage scores. *Theoretical Computer Science*, 657:38–47, 2017.

[36] Michael W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[37] Mohammadhossein Mohammadisiahroudi, Ramin Fakhimi, and Tamás Terlaky. Efficient use of quantum linear system algorithms in interior point methods for linear optimization. *arXiv preprint arXiv:2205.01220*, 2022.

[38] Ashley Montanaro. Quantum speedup of monte carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2181):20150301, 2015.

[39] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Studies in Applied and Numerical Mathematics. SIAM, 1993.

[40] Aditya Parulekar, Advait Parulekar, and Eric Price. L1 Regression with Lewis Weights Subsampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 49:1–49:21, 2021.

[41] James Renegar. A polynomial-time algorithm, based on Newton's method, for linear programming. *Mathematical Programming*, 40:59–93, 1988.

[42] James Renegar. *A Mathematical View of Interior-Point Methods in Convex Optimization*. MOS-SIAM Series on Optimization. SIAM, USA, 2001.

[43] Changpeng Shao. An improved quantum algorithm for low-rank rigid linear regressions with vector solution outputs. *arXiv preprint arXiv:2301.06107*, 2023.

[44] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[45] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90, 2004.

[46] Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, aug 2011.

[47] Pravin M. Vaidya. A new algorithm for minimizing convex functions over convex sets. *Mathematical Programming*, 73(3):291–341, June 1996.

[48] Pravin M. Vaidya and David S. Atkinson. A technique for bounding the number of iterations in path following algorithms. In *Complexity in Numerical Optimization*, pages 462–489. 1993.

A Quantum algorithm based on cutting plane method

The currently fastest cutting plane method to solve convex optimization problems is due to Lee, Sidford, and Wong [32]. For comparison, we state its complexity here and we show how to quantize it in the setting of linear programming.

Theorem A.1 (Informal, Lee-Sidford-Wong [32])**.** *Let $K \subseteq \mathbb{R}^d$ be a convex body contained in a (known) ball of radius R and let $\varepsilon > 0$. Then with $O(d \log(dR/\varepsilon))$ queries to a separation oracle for K and $\widetilde{O}(d^3)$ additional work, we can either find a point in K or prove that K does not contain a ball of radius ε.*

Together with a binary search on the objective value of $\min c^T x$ s.t. $Ax \geq b$ and a quantum implementation of a separation oracle (see below), this allows one to solve tall LPs quantumly in time

$$\widetilde{O}(\sqrt{n}dr + d^3).$$

The above mentioned quantum implementation of a (weak) separation oracle is a simple application of Grover's algorithm. Indeed, given a vector $x \in \mathbb{R}^d$, we need to determine if $Ax \geq b$. If not, then we need to return a hyperplane that separates x from the feasible region. We do so by using Grover's algorithm to search for a violated inequality: a violated inequality will separate x from the feasible region, whereas x is feasible if none are found. Verifying a single inequality $a_i^T x \geq b_i$ for $i \in [n]$ can be implemented using one row query to A, a single query to b, and time $\widetilde{O}(r)$. This implements a separation oracle in time $O(\sqrt{n}r)$.

B Omitted proofs about robustness of IPMs

We will use the fundamental theorem of calculus for the gradient.

Theorem B.1. *If $x, y \in D$, then*

$$g(y) - g(x) = \int_0^1 H(x + t(y - x))(y - x)\, dt.$$

Our analysis of approximate Newton steps builds on the usual analysis for exact Newton steps. We state here the key results for the exact case that we use.

Path-following requires us to optimize a self-concordant function, given an approximate minimizer. For this we can use Newton's method, which makes updates of the form

$$x_+ := x - H(x)^{-1} g(x) = x + n(x), \tag{10}$$

where $n(x)$ denotes the Newton step at x. The next theorem shows the two key properties of the (exact) Newton step: its size decreases essentially quadratically, and its size bounds the distance to the minimizer of f.

Theorem B.2 ((Exact) Newton steps [42, Thrm. 2.2.4-5])**.** *Assume f is self-concordant.*

- *If $\|n(x)\|_x < 1$, then $\|n(x_+)\|_{x_+} \leq \left(\frac{\|n(x)\|_x}{1-\|n(x)\|_x}\right)^2$.*

- *If $\|n(x)\|_x \le 1/4$ for some $x \in D$, then f has a minimizer z and*

$$\|x_+ - z\|_x \le \frac{3\|n(x)\|_x^2}{(1-\|n(x)\|_x)^3}.$$

Thus

$$\|x - z\|_x \le \|n(x)\|_x + \frac{3\|n(x)\|_x^2}{(1-\|n(x)\|_x)^3} \le (5/6)^2.$$

B.1 Omitted proofs of Section 6.2

We prove an extended version of Lemma 6.8.

Lemma B.3 (Long version of Lemma 6.8)**.** *Assume f is self-concordant, $x \in D$, and $Q(x)$ is such that $Q(x) \preceq H(x) \preceq CQ(x)$ for $C \ge 1$. Let $d(x) = -Q(x)^{-1}g(x)$ and assume $\tilde{d}(x)$ is such that $\|\tilde{d}(x) - d(x)\|_x \le \zeta$. Then for $0 < \delta < 1$ we have*

$$f(x + \delta\tilde{d}(x)) \le f(x) - \|\tilde{d}(x)\|_x^2 \left(\frac{\delta}{C} - \frac{\delta^2}{2(1-\delta\|\tilde{d}(x)\|_x)^2}\right) + \delta\|\tilde{d}(x)\|_x\|\tilde{d}(x) - d(x)\|_x,$$

and in particular, if $\delta\|\tilde{d}(x)\|_x \le 1/2$ for $\delta = 1/(4C)$ and $\zeta \le \frac{1}{C}\|\tilde{d}(x)\|_x$, then we have

$$f(x + \delta\tilde{d}(x)) \le f(x) - \frac{1}{16C^2}\|\tilde{d}(x)\|_x^2.$$

Proof of Lemma 6.8. For $0 < \delta < 1$ there exists some $\bar{\delta} \in [0, \delta]$ such that

$$\begin{aligned} f(x + \delta\tilde{d}(x)) &= f(x) + \delta\langle g(x), \tilde{d}(x)\rangle + \frac{1}{2}\delta^2\langle\tilde{d}(x), H(x + \bar{\delta}\tilde{d}(x))\tilde{d}(x)\rangle \\ &= f(x) + \delta\langle g(x), d(x)\rangle + \delta\langle g(x), \tilde{d}(x) - d(x)\rangle + \frac{1}{2}\delta^2\langle\tilde{d}(x), H(x + \bar{\delta}\tilde{d}(x))\tilde{d}(x)\rangle \end{aligned}$$

We upper bound the last three terms on the right hand side. For the first term, since $g(x) = -Q(x)d(x)$, we have $\langle g(x), d(x)\rangle = -\|d(x)\|_{Q(x)}^2 \le -\|d(x)\|_x^2/C$ where in the last inequality we use $Q(x) \succeq \frac{1}{C}H(x)$. For the second term we have

$$\begin{aligned} \langle g(x), \tilde{d}(x) - d(x)\rangle &= \langle d(x), \tilde{d}(x) - d(x)\rangle_{Q(x)} \\ &\le \|d(x)\|_{Q(x)}\|\tilde{d}(x) - d(x)\|_{Q(x)} \\ &\le \|d(x)\|_x\|\tilde{d}(x) - d(x)\|_x, \end{aligned}$$

where in the last inequality we use that $Q(x) \preceq H(x)$. Finally, for the third term, we have

$$\begin{aligned} \langle\tilde{d}(x), H(x + \bar{\delta}\tilde{d}(x))\tilde{d}(x)\rangle &= \langle\tilde{d}(x), H_x(x + \bar{\delta}\tilde{d}(x))\tilde{d}(x)\rangle_x \\ &= \|\tilde{d}(x)\|_x^2 + \langle\tilde{d}(x), (H_x(x + \bar{\delta}\tilde{d}(x)) - I)\tilde{d}(x)\rangle_x \\ &\le \|\tilde{d}(x)\|_x^2 + \|H_x(x + \bar{\delta}\tilde{d}(x)) - I\|_x\|\tilde{d}(x)\|_x^2 \\ &\le \|\tilde{d}(x)\|_x^2/(1 - \bar{\delta}\|\tilde{d}(x)\|_x)^2 \\ &\le \|\tilde{d}(x)\|_x^2/(1 - \delta\|\tilde{d}(x)\|_x)^2, \end{aligned}$$

where in the second inequality we used Theorem 6.4. Combining the three estimates concludes the proof of the first statement. For the second statement, assume $\delta\|d(x)\|_x \leq 1/2$ for $\delta = \frac{1}{4C}$, then we have

$$\frac{\delta}{C} - \frac{\delta^2}{2(1-\delta\|d(x)\|_x)^2} - \frac{\delta}{4C} \geq \frac{\delta}{C} - \frac{\delta^2}{2(1/2)^2} - \frac{\delta}{4C} \geq \frac{\delta}{C} - \frac{\delta}{2C} - \frac{\delta}{4C} = \frac{\delta}{4C} = \frac{1}{16C^2}.$$

□

Proof of Lemma 6.9. Since f is convex, we have for all $x, y \in D$ that

$$f(x) - f(y) \leq \langle g(x), x - y\rangle = \langle n(x), x - y\rangle_x.$$

We can further upper bound the right hand side using the Cauchy-Schwarz inequality:

$$\langle n(x), x - y\rangle_x \leq \|n(x)\|_x \|x - y\|_x.$$

We apply the above with $y = z$ and it thus remains to upper bound $\|x - z\|_x$. For this we use the second bullet in Theorem B.2: we have $\|x - z\|_x \leq \|n(x)\|_x + \frac{3\|n(x)\|_x^2}{(1-\|n(x)\|_x)^3}$. Combining the two upper bounds on $f(x) - f(z)$ concludes the proof. □