Quantum Speedups for Zero-Sum Games via Improved Dynamic Gibbs Sampling

Adam Bouland [1] **Yosheb Getachew** [1] **Yujia Jin** [1] **Aaron Sidford** [1] **Kevin Tian** [2]

Abstract

We give a quantum algorithm for computing an ϵ-approximate Nash equilibrium of a zero-sum game in an $m \times n$ payoff matrix with bounded entries. Given a standard quantum oracle for accessing the payoff matrix our algorithm runs in time $\widetilde{O}(\sqrt{m+n} \cdot \epsilon^{-2.5} + \epsilon^{-3})$ and outputs a classical representation of the ϵ-approximate Nash equilibrium. This improves upon the best prior quantum runtime of $\widetilde{O}(\sqrt{m+n} \cdot \epsilon^{-3})$ obtained by (van Apeldoorn & Gilyén, 2019) and the classical $\widetilde{O}((m+n) \cdot \epsilon^{-2})$ runtime due to (Grigoriadis & Khachiyan, 1995) whenever $\epsilon = \Omega((m+n)^{-1})$. We obtain this result by designing new quantum data structures for efficiently sampling from a slowly-changing Gibbs distribution.

1 Introduction

For a wide range of machine learning and numerical linear algebra problems, quantum algorithms (in certain parameter regimes) yield faster runtimes than classical counterparts (Biamonte et al., 2017).[1] Leveraging quantum algorithmic primitives, e.g. (Brassard et al., 2002; Harrow et al., 2009; Gilyén et al., 2019), these algorithms obtain runtimes which improve upon the dimension dependence of classical algorithms, but often at the cost of a worse dependence on the error tolerance and/or implicit access to the solution (e.g. query or sampling access for solution entries). Consequently, this paper is motivated by the following question.

What accuracy versus dimension-dependence tradeoffs are inherent in quantum optimization? What techniques improve this tradeoff?

We consider this question for the fundamental optimization problem of computing ϵ-approximate Nash equilibrium in zero-sum games. Our main result is an improved dependence on ϵ for quantum algorithms solving zero-sum games, which is very close to that of its classical counterpart. Further, we show that for our algorithms, obtaining a classical representation of the solution is obtainable at no additional asymptotic cost. Our work builds upon (van Apeldoorn & Gilyén, 2019; Li et al., 2019), which already took a large and important step towards answering the above question by designing quantum data structures for efficiently implementing algorithms for solving zero-sum games.

To obtain our result we provide improved quantum algorithms for solving a dynamic data structure problem of sampling from a slowly-changing Gibbs distribution. Such dynamic sampling problems arise as a natural component of stochastic gradient methods for solving zero-sum games. We obtain our speedups by improving a Gibbs sampling subroutine developed in (van Apeldoorn & Gilyén, 2019). We design a new dynamic quantum data structure performing the necessary sampling in time $\widetilde{O}(\epsilon^{-\frac{1}{2}})$, faster than the corresponding $\widetilde{O}(\epsilon^{-1})$ runtime achieved by (van Apeldoorn & Gilyén, 2019). Beyond the intrinsic utility of solving this problem, we hope our improved Gibbs sampler showcases algorithmic insights that can be gleaned by seeking improved error dependencies for quantum optimization algorithms. Moreover, we hope this work encourages the study of quantum data structures for efficient optimization.

1.1 Zero-sum games

For matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ its associated *zero-sum game* is the pair of equivalent optimization problems

$$\min_{u \in \Delta^m} \max_{v \in \Delta^n} u^\top \mathbf{A} v = \max_{v \in \Delta^n} \min_{u \in \Delta^m} u^\top \mathbf{A} v,$$
$$\text{where } \Delta^k := \{x \in \mathbb{R}^k_{\geq 0} : \textstyle\sum_{i \in [k]} x_i = 1\}.$$

In such a game, we refer to $\mathbf{A}$ as the *payoff matrix* and view the m and n-dimensional simplices, i.e. Δ^m and Δ^n, as the

[1]Stanford University, Stanford, CA, USA. [2]Microsoft Research, Redmond, WA, USA. Correspondence to: Yosheb Getachew <yoshebg@stanford.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]Note that quantifying the end-to-end speedups obtained by these methods can be subtle due to I/O overheads, different access models (Aaronson, 2015), and classical de-quantization algorithms (Tang, 2019; Chia et al., 2020; Gharibian & Le Gall, 2022).

space of distributions over $[m]$ and $[n]$ respectively. From this perspective $u^\top \mathbf{A} v$, known as *payoff* or *utility* of (u, v), is the expected value of $\mathbf{A}_{ij}$ when sampling $i \in [m]$ and $j \in [n]$ independently from the distributions corresponding to u and v. Thus, a zero-sum game models a two-player game where one player seeks to minimize the payoff while, simultaneously, the other seeks to maximize it.

We consider the canonical problem of computing an approximate Nash equilibrium of a zero-sum game. Given the payoff matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ we call a pair $(u, v) \in \Delta^m \times \Delta^n$ an ϵ-*approximate Nash equilibrium (NE)* for $\epsilon \in \mathbb{R}_{>0}$ if

$$\left(\max_{v' \in \Delta^n} u^\top \mathbf{A} v'\right) - \left(\min_{u' \in \Delta^m} (u')^\top \mathbf{A} v\right) \leq \epsilon.$$

We assume the payoff matrix $\mathbf{A}$ and the error-tolerance are given as input to an algorithm, and $\|\mathbf{A}\|_{\max} \leq 1$, i.e. the largest entry of $\mathbf{A}$ has magnitude at most 1 (without loss of generality, by rescaling $\mathbf{A} \leftarrow \|\mathbf{A}\|_{\max}^{-1} \mathbf{A}$ and $\epsilon \leftarrow \|\mathbf{A}\|_{\max}^{-1} \epsilon$). Our main goal is to design improved zero-sum game solvers, i.e. algorithms computing ϵ-approximate NEs.

Zero-sum games are foundational to optimization, economics, and computer science. Solving zero-sum games is a natural formulation of linear programming (LP) and thus it is a prominent testbed for new optimization techniques. There have been numerous advances in the computational complexity of solving zero-sum games under various assumptions on problem parameters (see Section 1.3). Recent advancements in interior point methods (IPMs), e.g. (van den Brand, 2020; van den Brand et al., 2021) and references therein (discussed in Section 1.3), solve zero sum-games in time $\widetilde{O}(mn + \min(m, n)^{2.5})$ or $\widetilde{O}((m + n)^\omega)$, where $\omega < 2.372$ is the current matrix multiplication constant (Duan et al., 2023).[2] In this paper, our focus is on sublinear-time algorithms for approximating NEs.

A well-known algorithm by (Grigoriadis & Khachiyan, 1995) achieves a runtime of $\widetilde{O}((m + n) \cdot \epsilon^{-2})$, the state-of-the-art sublinear runtime amongst classical algorithms, without further problem assumptions. Recently it has been shown that quantum algorithms can yield striking runtime improvements for solving zero-sum games and their generalizations (Li et al., 2019; van Apeldoorn & Gilyén, 2019; Li et al., 2021). In particular, Li, Chakrabati and Wu (Li et al., 2019) gave a quantum algorithm for zero-sum games in time $\widetilde{O}(\sqrt{m + n} \cdot \epsilon^{-4})$, and simultaneously van Apeldoorn and Gilyen (van Apeldoorn & Gilyén, 2019) obtained a runtime of $\widetilde{O}(\sqrt{m + n} \cdot \epsilon^{-3})$. These algorithms yield a quadratic improvement in the dimension dependence of the best classical algorithm, at the cost of a higher ϵ dependence.

The algorithms of (Li et al., 2019; van Apeldoorn & Gilyén, 2019; Li et al., 2021) operate using a standard quantum oracle for querying entries of $\mathbf{A}$ (formally stated in Section 1.4). We focus on discussing the algorithm of (van Apeldoorn & Gilyén, 2019), as our focus is error dependence. The (van Apeldoorn & Gilyén, 2019) algorithm generalizes the classical algorithm of (Grigoriadis & Khachiyan, 1995), and improves it by speeding up a key dynamic Gibbs sampling subroutine. As we survey in Section 2, van Apeldoorn and Gilyen give a quantum data structure to efficiently perform this sampling in time quadratically faster in the dimension, the core of their algorithmic speedup.

Our result. We give a new quantum algorithm for solving zero-sum games, improving the prior state-of-the-art quantum runtime, due to (van Apeldoorn & Gilyén, 2019).

Theorem 1.1 (informal, see Theorem 2.3)**.** *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $\|\mathbf{A}\|_{\max} \leq 1$, and $\epsilon \in (0, 1)$. Given a quantum oracle for $\mathbf{A}$ (cf. Section 1.4), there is an $\widetilde{O}(\sqrt{m + n} \cdot \epsilon^{-2.5} + \epsilon^{-3})$ time algorithm which yields a classical output $(u, v) \in \Delta^m \times \Delta^n$ that is an ϵ-approximate NE with high probability.*

Our new algorithm simultaneously improves the best known quantum (van Apeldoorn & Gilyén, 2019) and classical (Grigoriadis & Khachiyan, 1995) algorithms in the parameter regime where IPMs do not dominate sublinear algorithms. In particular, it is faster than the classical $\widetilde{O}((m + n) \cdot \epsilon^{-2})$ runtime of (Grigoriadis & Khachiyan, 1995) whenever $\epsilon^{-1} = \widetilde{O}(m + n)$, which includes the regime where (Grigoriadis & Khachiyan, 1995) offers advantages over the $\widetilde{O}((m + n)^\omega)$ runtime of the (van den Brand, 2020) IPM, as $\omega < 3$. This is in contrast to (van Apeldoorn & Gilyén, 2019), which does not achieve an improvement upon (Grigoriadis & Khachiyan, 1995) in the full parameter range where sublinear algorithms are currently preferable to IPMs. For example, when $m \approx n$ and (up to logarithmic factors) $\epsilon \in [n^{-c}, n^{-\frac{1}{2}}]$ where $c = \frac{1}{2}(\omega - 1)$, the rate of (Grigoriadis & Khachiyan, 1995) is favorable to that of (van Apeldoorn & Gilyén, 2019) and state-of-the-art IPMs (van den Brand, 2020; Cohen et al., 2021).[3]

1.2 Dynamic Gibbs sampling

We obtain our improved zero-sum game solver by producing a new, faster quantum data structure to perform the Gibbs sampling used in the (van Apeldoorn & Gilyén, 2019) algorithm, of possible independent interest. Gibbs sampling is a fundamental algorithmic primitive — the basic task is, given vector $v \in \mathbb{R}^n$, sample proportionally to $\exp(v)$. It is used as a subroutine in many quantum and classical optimization algorithms, e.g. (Brandao & Svore, 2017) and follow-ups. Quantum amplitude estimation can speed up this task, by boosting the acceptance probability of rejection sampling

[2]We $\widetilde{O}$ to hide polylogarithmic dependences, and "with high probability" to indicate a polylogarithmic dependence on the failure probability; see Section 1.4 for a more detailed statement.

[3]There is evidence that $\omega = 2$ cannot be achieved with current techniques, e.g. (Alman, 2021).

subroutines. This strategy was used by (van Apeldoorn & Gilyén, 2019), and is discussed in Section 2.2.

By storing partial information about the Gibbs distribution, namely an efficiently-computable overestimate to its entries which remains valid across many consecutive iterations, we obtain an improved dynamic Gibbs sampler, which we also provide a detalied overview of in Section 2.2.

We now define our notion of an approximate Gibbs sampler, and state the dynamic sampling problem we consider, arising naturally in sublinear-time zero-sum game algorithms.

Definition 1.2 (Approximate Gibbs oracle). *For $v \in \mathbb{R}^n$, its associated* Gibbs distribution *is $p_v \in \Delta^n$ such that $[p_v]_i \propto \exp(v_i)$, $\forall i \in [n]$. We say $\mathcal{O}_v^{\text{gibbs}}$ is a δ-approximate Gibbs oracle if it samples from $\tilde{p} \in \Delta^n$ with $\|\tilde{p} - p_v\|_1 \leq \delta$.*

Problem 1 (Sampling maintenance). *Let $\eta > 0$, $\delta \in (0,1)$, and suppose we have a quantum oracle for $\mathbf{A} \in \mathbb{R}^{m\times n}$. Consider a sequence of T* Update *operations to a dynamic vector $x \in \mathbb{R}^m_{\geq 0}$, of the form $x_i \leftarrow x_i + \eta$ for $i \in [m]$. In the* sampling maintenance problem*, in amortized $\mathcal{T}_{\text{update}}$ time per* Update *we must maintain a δ-approximate Gibbs oracle, $\mathcal{O}_{\text{samp}}$, for $\mathbf{A}^\top x$ queryable in worst-case time $\mathcal{T}_{\text{samp}}$.*

Our result. We provide a quantum algorithm for Problem 1, which improves upon the runtime implied by the corresponding subroutine in (van Apeldoorn & Gilyén, 2019).

Theorem 1.3 (informal, see Theorem 2.2). *There is a quantum algorithm which solves Problem 1 with high probability with $\max(\mathcal{T}_{\text{samp}}, \mathcal{T}_{\text{update}}) = \widetilde{O}\left(\sqrt{n}\cdot T\eta^{1.5}\right)$ and an initialization cost of $\widetilde{O}\left(\eta^3 T^3\right)$.*

Theorem 1.3 improves upon the solution to the sampling maintenance Problem 1 implied by (van Apeldoorn & Gilyén, 2019) by a $\eta^{-\frac{1}{2}}$ factor; in the setting of the (Grigoriadis & Khachiyan, 1995) solver, where $T = \widetilde{O}(\epsilon^{-2})$ and $\eta = \Theta(\epsilon)$, this is an $\epsilon^{-\frac{1}{2}}$-factor. Our improvement is obtained by storing a hint consisting of a vector which overestimates the true Gibbs distribution, and an approximate normalization factor, which are infrequently updated. Our maintained hint satisfies the desirable properties that: (i) it remains valid for a batch of consecutive iterations, and (ii) the degree of overestimation is bounded. The former property ensures a fast amortized update time, and the latter ensures a fast sample time by lower bounding the acceptance probability of our quantum rejection sampler. Our strategy for maintaining improved hints is to call our sampling access to estimate large entries of the Gibbs distribution, and to exploit stability under the setting of Problem 1. We discuss our sampler in more detail and compare it with previous methods for solving Problem 1 in Section 2.2.

The initialization cost of Theorem 1.3 is due to the current state-of-the-art in numerically stable implementations of the quantum singular value transformation (SVT) framework of (Gilyén et al., 2019). This cost is also the cause of the additive $\widetilde{O}(\epsilon^{-3})$ term in Theorem 1.1. We discuss this cost in Appendix E; improvements to numerically stable implementations of (Gilyén et al., 2019) would also be reflected in the runtimes of Theorems 1.1 and 1.3.

1.3 Related work

Quantum optimization and machine learning. A wide array of quantum algorithms for optimization and machine learning makes use of fundamental algorithmic primitives such as amplitude amplification (Brassard et al., 2002), the HHL algorithm (Harrow et al., 2009), and the quantum singular value transformation (Gilyén et al., 2019). For example, a number of works gave HHL-based algorithms for machine learning tasks such as PCA (Lloyd et al., 2014), SVMs (Rebentrost et al., 2014), and recommendation systems (Kerenidis & Prakash, 2016). For more details see the survey article of (Biamonte et al., 2017).

Most relevant to our work are quantum algorithms for optimization. For example, Brandao and Svore (Brandao & Svore, 2017) gave a quantum semidefinite programming (SDP) solver based on the Arora-Kale algorithm (Arora & Kale, 2007), later improved by (Van Apeldoorn et al., 2020). There have also been quantum IPM-based methods for LPs and SDPs (Kerenidis & Prakash, 2020). Further, quantum algorithms for general convex optimization have been developed (Chakrabarti et al., 2020; van Apeldoorn et al., 2020), making use of Jordan's algorithm for fast gradient estimation (Jordan, 2005; Gilyén et al., 2019).

Regarding zero-sum games, in addition to the works (van Apeldoorn & Gilyén, 2019; Li et al., 2019) on ℓ_1-ℓ_1 games (where both players are ℓ_1-constrained), there have been several works considering different variants. For example (Li et al., 2019) gave quantum algorithms for ℓ_2-ℓ_1 games with quadratic improvement on the dimension, later extended by (Li et al., 2021) to ℓ_q-ℓ_1 games with $q \in (1, 2]$.

Zero-sum games. Zero-sum games are a canonical modeling tool in optimization, economics and machine learning (Neumann, 1928). The classic extragradient method (Nemirovski, 2004; Nesterov, 2007) computes an ϵ-approximate NE in $\widetilde{O}(mn \cdot \epsilon^{-1})$ time and, as discussed, the stochastic mirror descent method of (Grigoriadis & Khachiyan, 1995) obtains the same accuracy in time $\widetilde{O}((m+n)\cdot\epsilon^{-2})$. An improved $\Omega(mn)$ runtime was recently obtained by (Carmon et al., 2019b) using variance reduction (see Table 1). Improved runtimes are available under more fine-grained assumptions on the matrix $\mathbf{A}$, such as sparsity or continuous analogs of sparsity (Carmon et al., 2020). It is worth noting that the sampling strategies employed in this paper are in a similar spirit as the variance reduction strategy employed by (Carmon et al., 2019b); each infrequently tracks partial information from previous iterates to achieve com-

Table 1: **Algorithms for ϵ-approximate NE of zero-sum games.** Hides logarithmic factors, assumes $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $\|\mathbf{A}\|_{\max} \leq 1$.

Method	Query model	Total runtime
interior point method (Cohen et al., 2021)	classical	$\max(m, n)^{\omega}$
interior point method (van den Brand et al., 2021)	classical	$mn + \min(m, n)^{2.5}$
extragradient (Nemirovski, 2004; Nesterov, 2007)	classical	$mn \cdot \epsilon^{-1}$
stochastic mirror descent (SMD) (Grigoriadis & Khachiyan, 1995)	classical	$(m + n) \cdot \epsilon^{-2}$
variance-reduced SMD (Carmon et al., 2019b)	classical	$mn + \sqrt{mn(m + n)} \cdot \epsilon^{-1}$
(van Apeldoorn & Gilyén, 2019)	quantum	$\sqrt{m + n} \cdot \epsilon^{-3}$
Theorem 1.1 (**our work**)	quantum	$\sqrt{m + n} \cdot \epsilon^{-2.5} + \epsilon^{-3}$

Table 2: **Solutions to Problem 1, $T = \epsilon^{-2}$, $\eta = \epsilon$.** Hides logarithmic factors.

Method	Query model	$\mathcal{T}_{\text{samp}}$	$\mathcal{T}_{\text{update}}$
explicit updates (Grigoriadis & Khachiyan, 1995)	classical	1	$m + n$
max-based rejection sampling (van Apeldoorn & Gilyén, 2019)	quantum	$\sqrt{m + n} \cdot \epsilon^{-1}$	$\sqrt{m + n} \cdot \epsilon^{-1}$
Theorem 1.3 (**our work**)	quantum	$\sqrt{m + n} \cdot \epsilon^{-\frac{1}{2}}$	$\sqrt{m + n} \cdot \epsilon^{-\frac{1}{2}}$

plexity improvements. In the case of variance reduction in (Carmon et al., 2019b), this partial information is used to obtain lower-variance estimators for lower iteration counts; in our case, it is used to boost the success rate of rejection sampling techniques. It is an interesting direction of future work to see if the techniques of this paper could be combined with the variance reduction tools used in (Carmon et al., 2019b) to obtain improved runtimes. Finally, we note that (Grigoriadis & Khachiyan, 1995) also offers runtime improvements under a sparsity assumption, as does (van Apeldoorn & Gilyén, 2019) when their are certain bounds on the sparsity-to-accuracy ratio. We focus on the general setting in this paper (without further sparsity assumptions), but believe that developing techniques achieving more fine-grained rates in terms of sparsity, or numerical sparsity as in (Carmon et al., 2020), is another promising future direction.

In parallel, a long line of research improving IPMs for solving linear programming (Karmarkar, 1984; Renegar, 1988; Lee & Sidford, 2014; 2019; van den Brand et al., 2020; Jiang et al., 2021) has led to a number of different zero-sum game solvers with polylogarithmic runtime dependencies on the problem accuracy ϵ. The current state-of-the-art variants of IPMs are (Cohen et al., 2021) and (van den Brand et al., 2021), which achieve runtimes of $\widetilde{O}(\max(m, n)^{\omega})$ and $\widetilde{O}(mn + \min(m, n)^{2.5})$ respectively: see Table 1. Finally, for strongly polynomial runtimes, which are outside our scope, we refer readers to (Dadush et al., 2020).

Future work. Theorem 1.1's ϵ dependence is within an $\epsilon^{-\frac{1}{2}}$ factor of classical counterparts. To our knowledge, removing this $\epsilon^{-\frac{1}{2}}$ overhead would represent the first quantum algorithm for a natural optimization problem which improves upon classical counterparts across all parameters.

Both our work and (van Apeldoorn & Gilyén, 2019) solve Problem 1 by leveraging a powerful polynomial approximation-based technique developed in (Gilyén et al., 2019), known as the quantum singular value transform (QSVT). In both cases, QSVT is used with a polynomial of degree $\widetilde{O}(\epsilon^{-1})$. We note that in closely-related classical settings (discussed in (Sachdeva & Vishnoi, 2014)), Chebyshev polynomial-based approximations yield a quadratically smaller degree. However, a boundedness requirement (due to the spectra of quantum gates) prevents straightforwardly applying these constructions within QSVT. Sidestepping this barrier is a natural avenue for improvement. More generally, establishing optimal oracle query complexities of dynamic Gibbs sampling (Problem 1) and solving zero-sum games are key problems left open by our work. These oracle complexity questions are potentially more approachable than establishing tight time complexity characterizations.

1.4 Preliminaries

General notation. $\widetilde{O}$ hides logarithmic factors in problem dimensions (denoted m and n), target accuracies (denoted ϵ), and failure probabilities (denoted α). When discussing Problem 1, we additionally use $\widetilde{O}$ to hide logarithmic factors in η, T. For all $i \in [n]$ we let $e_i \in \mathbb{R}^n$ denote the i^{th} standard basis vector for $i \in [n]$ when n is clear. $\|\cdot\|_p$ denotes the ℓ_p norm of a vector. For $\mathbf{A} \in \mathbb{R}^{m\times n}$, its i^{th} row and j^{th} column are respectively $\mathbf{A}_{i:}, \mathbf{A}_{:j}$. For $v \in \mathbb{R}^n$, $\mathbf{diag}(v)$ is the diagonal $n \times n$ matrix with v as the diagonal. Conjugate transposes of $\mathbf{A}$ are denoted $\mathbf{A}^*$; when the matrix is real we use $\mathbf{A}^\top$. The all-ones and all-zeros vectors of dimension n are $\mathbf{1}_n$ and $\mathbf{0}_n$. Finally, throughout $a := \lceil \log_2 m \rceil$ and $b := \lceil \log_2 n \rceil$, so $[m] \subseteq [2^a]$ and $[n] \subseteq [2^b]$.

Computation models. We assume entries of $\mathbf{A}$ are w-bit reals for $w = O(\log(mn))$, and work in the word RAM model where w-bit arithmetic operations take $O(1)$ time; for simplicity, we assume mathematical operations such as trignometric functions and radicals can also be implemented exactly for w-bit words in $O(1)$ time. Throughout, "quantum states" mean unit vectors, and "quantum gates" or "oracles" $\mathcal{O}$ mean unitary matrices. We follow standard notation and identify a standard basis vector e_i for $i \in [n]$ with $|i\rangle$, an a-qubit state, in which i is represented in binary (i.e. more formally, $|i\rangle = |\text{bin}(i)\rangle$, and bin is omitted for brevity). We consider the standard model of quantum access to oracles, in which the oracle $\mathcal{O}$, which is defined by its operation on $|s\rangle$ for all $\{0,1\}^*$-valued s (where length is clear from context), can be queried in superposition. If $\mathcal{O}$ is queried on $|v\rangle := \sum_s \alpha_s |s\rangle$, the result is $\mathcal{O}|v\rangle = \sum_s \alpha_i(\mathcal{O}|s\rangle)$. We use $|g\rangle$, $|g'\rangle$, etc. (when clear from context) to denote arbitrary sub-unit vectors, which represent garbage states (unused in computations). The tensor product of states $|u\rangle$ and $|v\rangle$ on a and b qubits is denoted $|u\rangle|v\rangle$, an $(a+b)$-qubit state. The runtime of a quantum circuit is its maximum depth (in arithmetic gates on w-bit words).

Access model. Throughout the paper, we assume a standard quantum oracle for accessing $\mathbf{A}$ (recall $\|\mathbf{A}\|_{\max} \le 1$). In particular, by a quantum oracle for $\mathbf{A}$ we mean an oracle $\mathcal{O}_{\mathbf{A}}$ which, when queried with $|i\rangle|j\rangle|s\rangle$ for $i \in [m], j \in [n], s \in \{0,1\}^w$, reversibly writes $\mathbf{A}_{ij}$ (in binary) to the third register in $O(1)$ time, i.e. $\mathcal{O}_{\mathbf{A}}|i\rangle|j\rangle|s\rangle = |i\rangle|j\rangle|s \oplus \mathbf{A}_{ij}\rangle$ where $\oplus$ is bitwise mod-2 addition.

Given a quantum oracle for $\mathbf{A}$, with two queries, by standard constructions one can construct an oracle which places the value in the amplitude of the state rather than the register itself. More formally, one can construct[4] an $\mathcal{O}'_{\mathbf{A}}$, which operates as: for $(i,j) \in [m] \times [n]$,

$$\mathcal{O}'_{\mathbf{A}}|0\rangle|i\rangle|j\rangle = \sqrt{\mathbf{A}_{ij}}|0\rangle|i\rangle|j\rangle + \sqrt{1 - |\mathbf{A}_{ij}|}|1\rangle|g\rangle.$$

It is standard to (using ancilla qubits to store the output register where $\mathbf{A}_{ij}$ is written) construct $\mathcal{O}'_{\mathbf{A}}$ from $\mathcal{O}_{\mathbf{A}}$ under our classical model of computation, see e.g. (Grover & Rudolph, 2002), and so we assume direct access to $\mathcal{O}'_{\mathbf{A}}$.

[4]This follows by calling $\mathcal{O}_{\mathbf{A}}$ to obtain $\mathbf{A}_{ij}$ in binary (as a signed number $\in [0,1]$), adding an ancilla qubit, computing the rotation angle needed on that ancilla, applying controlled rotation gates to an ancilla using that angle, then calling $\mathcal{O}_{\mathbf{A}}$ a second time to uncompute $\mathbf{A}_{ij}$. See e.g. (Grover & Rudolph, 2002) for details.

Organization. In Section 2, we give a brief technical overview of the core components of our algorithm used to prove Theorem 1.1: the stochastic gradient method our method is built on, and an efficient quantum implementation of a key subroutine using a new dynamic Gibbs sampler. Finally in Section 3 we give our new quantum sampler, and prove Theorem 1.3. We aim to give a self-contained description of our algorithm in Section 2 to improve readability for readers with an optimization background unfamiliar with quantum computing, and vice versa. In particular, we abstract away the core optimization machinery (stochastic mirror descent) and quantum machinery (quantum SVT) developed in prior work into Propositions 2.1 and 2.5, and focus on using these statements black-box to build a faster algorithm. We defer proofs to Appendices A to C.

2 Overview of approach

We now overview our approach for obtaining our main results: an improved quantum runtime for solving zero-sum games (Theorem 2.3) and an improved quantum data structure for dynamic Gibbs sampling (Theorem 2.2). In Section 2.1, we state Algorithm 1, the optimization framework we use to solve zero-sum games, generalizing the algorithm of (Grigoriadis & Khachiyan, 1995). We state its guarantees in Proposition 2.1, proven in Appendix A. Algorithm 1 assumes access to an approximate Gibbs oracle (Definition 1.2) for sampling from the distributions stated in Problem 1. Much of our work is devoted to obtaining an efficient quantum implementation of such an oracle (Theorem 2.2), used to prove Theorem 2.3 at the end of Section 2.1. In Section 2.2, we overview the main technical innovation of this paper, an improved solution to Problem 1. The only quantum components of our algorithm are abstracted away by Proposition 2.5, proven in Appendix B.

2.1 Solving matrix games via Gibbs sampling

Our proof of Theorem 2.3 uses an efficient implementation of the framework in Algorithm 1, based on stochastic mirror descent. In specifying Algorithm 1, we recall our earlier Definition 1.2, which captures the approximate sampling access we require for Algorithm 1's execution.

The main skeleton of Algorithm 1 (Lines 5-6) using exact oracles is identical to the method of (Grigoriadis & Khachiyan, 1995). We build upon (Grigoriadis & Khachiyan, 1995) in the following three ways: we (1) tolerate total variation error in the sampling procedure via δ-approximate Gibbs oracles, (2) provide a high-probability guarantee on the duality gap using martingale arguments, and (3) subsample the output to obtain a sparse solution yielding a comparable duality gap.

Algorithm 1: MatrixGameSolver(δ, η, T)

```
1 Input: $\mathbf{A} \in \mathbb{R}^{m\times n}$, desired accuracy $\epsilon \in (0,1)$, $\delta$-approximate Gibbs oracles for the (dynamic) vectors $-\mathbf{A}^\top x_t$ and $\mathbf{A}y_t$
2 Parameters: Gibbs sampler parameter $\delta \in (0,1)$, step size $\eta > 0$, iteration count $T$
3 Initialize $\hat{u} \leftarrow \mathbf{0}_m$, $\hat{v} \leftarrow \mathbf{0}_n$, $x_0 \leftarrow \mathbf{0}_m$, and $y_0 \leftarrow \mathbf{0}_n$
4 for $t = 0$ to $T-1$ do
5     Independently sample $j_t, j_t' \in [n]$ using $\mathcal{O}^{\text{gibbs}}_{-\mathbf{A}^\top x_t}$ and $i_t, i_t' \in [m]$ using $\mathcal{O}^{\text{gibbs}}_{\mathbf{A}y_t}$
6     Update $y_{t+1} \leftarrow y_t + \eta e_{j_t}$ and $x_{t+1} \leftarrow x_t + \eta e_{i_t}$
7     Update $\hat{u} \leftarrow \hat{u} + \frac{1}{T} e_{i_t'}$ and $\hat{v} \leftarrow \hat{v} + \frac{1}{T} e_{j_t'}$
8 return $(\hat{u}, \hat{v})$
```

Several of these improvements have appeared previously in the literature. For example, an approximation-tolerant stochastic gradient method was given in (Carmon et al., 2020), and our proofs of the high-probability guarantees are based on arguments in (Allen-Zhu & Li, 2017; Carmon et al., 2019a). For completeness we give a self-contained proof of the following guarantee in Appendix A.

Proposition 2.1. *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ satisfy $\|\mathbf{A}\|_{\max} \le 1$ and $\epsilon, \alpha \in (0,1)$. Let $\delta \le \frac{\epsilon}{20}$, $\eta = \frac{\epsilon}{60}$, and $T = \Theta(\epsilon^{-2}\log\frac{mn}{\alpha})$ for an appropriate constant. With probability $\ge 1-\alpha$, Algorithm 1 outputs an ϵ-approximate NE for $\mathbf{A}$.*

Given Proposition 2.1, we simply need to efficiently implement the Gibbs sampling in Line 5. As introduced in Section 1, Problem 1 describes a dynamic approximate Gibbs oracle sampling problem sufficient for this task. By combining Proposition 2.1 with the following Theorem 2.2 (our solution to Problem 1), we prove our main result.

Theorem 2.2. *Let $\alpha \in (0,1)$, $\delta \le \eta$. Given a quantum oracle for $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $\|\mathbf{A}\|_{\max} \le 1$, we can solve Problem 1 with probability $\ge 1-\alpha$, and $\max(\mathcal{T}_{\text{samp}}, \mathcal{T}_{\text{update}}) = O(1 + \sqrt{n}T\eta\log^4\frac{mn}{\delta}\cdot(\sqrt{\eta\log\frac{n\eta T}{\alpha}} + \eta\log\frac{n\eta T}{\alpha}))$, and an initialization cost of $O(\eta^3T^3\log^4\frac{n\eta T}{\delta} + \log^7\frac{n\eta T}{\delta})$.*

Theorem 2.3. *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ satisfy $\|\mathbf{A}\|_{\max} \le 1$, and let $\epsilon, \alpha \in (0,1)$. Given a quantum oracle for $\mathbf{A}$, there is an algorithm returning a classical output $(u,v) \in \Delta^m \times \Delta^n$ that is an ϵ-approximate NE for $\mathbf{A}$ with probability $\ge 1-\alpha$ in time, for $\ell := \log\frac{mn}{\epsilon}$ and $\lambda := \log\frac{mn}{\alpha\epsilon}$,*

$$O\left(\frac{\sqrt{m+n}}{\epsilon^{2.5}}\ell^4\lambda^{2.5} + \frac{\sqrt{m+n}}{\epsilon^2}\ell^4\lambda^3 + \frac{1}{\epsilon^3}\ell^7\right).$$

Proof. We apply two instances of Theorem 2.2 to implement the δ-approximate Gibbs oracle for $-\mathbf{A}^\top x_t$ and $\mathbf{A}y_t$, to implement each iteration of Algorithm 1 in amortized $O(1 + \mathcal{T}_{\text{samp}} + \mathcal{T}_{\text{update}})$ time. Using the settings of parameters T, η in Proposition 2.1 and setting $\delta = \Theta(\epsilon)$, which suffices for Algorithm 1 and Theorem 2.2, we have $\max(\mathcal{T}_{\text{samp}}, \mathcal{T}_{\text{update}}) = O(\frac{\sqrt{m+n}}{\epsilon}\ell^4\lambda\cdot(\epsilon\lambda + \sqrt{\epsilon\lambda}))$. The conclusion follows since Algorithm 1 costs $O(T\cdot(1 + \mathcal{T}_{\text{samp}} + \mathcal{T}_{\text{update}}))$. The additive runtime term comes from the cost of stably implementing a quantum circuit required in the use of Theorem 2.2, discussed in Appendix E. □

We note despite the extra additive term, the runtime proven in Theorem 2.3 improves over the classical $\widetilde{O}((m+n)\cdot\epsilon^{-2})$ runtime of (Grigoriadis & Khachiyan, 1995) whenever $\epsilon^{-1} = \widetilde{O}(m+n)$.

2.2 Dynamic sampling maintenance via hints

We now overview our proof of Theorem 2.2 which proceeds in two steps. First, we reduce sampling maintenance (Problem 1) to a problem we call *hint maintenance*. This latter problem is a specialization of the sampling maintenance problem where suitable advice, the *hint*, is provided. Second, we solve hint maintenance as required by Proposition 2.5 in Theorem 2.2, by recursively calling Proposition 2.5, allowing us to maintain hints of suitable quality.

Reducing to hint maintenance. First, we introduce a data structure for maintaining the x variable in Problem 1, which was used crucially in (van Apeldoorn & Gilyén, 2019) for dynamic Gibbs sampling. This data structure allows efficient queries to subsets of the coordinates of x.

Lemma 2.4 (Sampler tree). *Let $\eta \in \mathbb{R}_{\ge 0}$ and $m \in \mathbb{N}$. There is a classical data structure,* SamplerTree*, supporting a tree on $O(m)$ nodes such that $[m]$ corresponds to leaves, with the following operations.*

- Init(m, η_{fixed})*: initialize $x \leftarrow \mathbf{0}_m$ and $\eta \leftarrow \eta_{\text{fixed}}$*
- Update(i)*: $x_i \leftarrow x_i + \eta$*
- SubtreeSum(v)*: return $\sum_{i \text{ in subtree of } v} x_i$*

The total runtime of T calls to Update *is $O(T\log m)$, and calls to* SubtreeSum *cost $O(1)$.*

An implementation of SamplerTree based on propagating subtree sums upon updates is a standard classical data structure. Next, we state our first building block towards solving Problem 1, which can be thought of as quantum sampling with a hint. We defer its proof to Appendix B, as it is based on generalizing dynamic block-encoding strategies with bounded-degree polynomial approximations, pioneered by (Gilyén et al., 2019; van Apeldoorn & Gilyén, 2019).

Proposition 2.5. *Let $x \in \mathbb{R}^m_{\ge 0}$ correspond to an instance of* SamplerTree*, and $\beta \ge \|x\|_1$. Let p be the Gibbs distribution associated with $\mathbf{A}^\top x$, let $Z := \sum_{j\in[n]}\exp([\mathbf{A}^\top x]_j)$ and $\widetilde{Z} \in [Z, CZ]$ for some $C \ge 1$. Finally, let $q \in \mathbb{R}^n$ have* *entries classically queriable in $O(1)$ time, satisfy $q \geq p$ entrywise, $q_j \in [\frac{\delta}{n}, 1]$ for all $j \in [n]$, and $\|q\|_1 = \rho$. Suppose $\widetilde{Z}$, C, ρ, and β are explicitly known. Given a quantum oracle for $\mathbf{A} \in \mathbb{R}^{m \times n}$ (defined in Section 1.4) with $\|\mathbf{A}\|_{\max} \leq 1$, we can implement a δ-approximate Gibbs oracle which has query cost $O(\sqrt{\rho C} \cdot \beta \log^4 \left(\frac{Cmn}{\delta}\right))$. The total additional cost incurred if x undergoes T Update calls which preserve the invariants on $\widetilde{Z}, C, \rho, \beta$ is $O(T \log m)$.*

Proposition 2.5 makes use of an overestimating hint vector q and approximate normalization constant $\widetilde{Z}$, which we collectively call the *hint*. The acceptance probability of our rejection sampling is governed by two primary parameters: $\rho = \|q\|_1$, which reflects the degree of overestimation (and can be thought of as a hint quality), and $C \geq 1$, which reflects our inability to accept with probability $\frac{p_j}{q_j}$ when p is implicit (which can be thought of as a normalization quality). In particular, the rejection sampling scheme used in Proposition 2.5 will instead accept with probability $\frac{p_j}{Cq_j}$.[5]

We elaborate briefly on the implementation of Proposition 2.5 (for details, see Appendix B). We follow notation of Proposition 2.5, and let $w := \mathbf{A}^\top x$ so the unnormalized Gibbs distribution is $\exp(w)$, and $p = \frac{\exp(w)}{Z}$. Proposition 2.5 is a rejection sampler which first loads the hint q into superposition, and then filters. Our scheme has the form

$$\text{sample } j \sim \frac{q}{\rho}, \text{ and accept with probability } \frac{p_j}{Cq_j}, \quad (1)$$

which results in an accepted sample with probability $\approx \frac{1}{\rho C}$, and hence requires $\approx \sqrt{\rho C}$ trials to succeed after applying quantum amplitude amplification, a generalization of Grover search (Brassard et al., 2002).[6] The latter filtering step is implemented using appropriate block-encoding technology.

This discussion suggests that the hint and normalization qualities, parameterized by ρ and C, are crucial in controlling the acceptance probability of our scheme. Concretely, in our applications of Proposition 2.5, $\beta = \eta T = \widetilde{O}(\frac{1}{\epsilon})$, which is the bound on the ℓ_1 norm of the x_t and y_t iterates in Algorithm 1 under the parameter settings of Proposition 2.1. Overall, the cost of implementing an approximate Gibbs oracle is then (up to logarithmic factors) $\sqrt{\rho C} \cdot \frac{1}{\epsilon}$. Proposition 2.5 hence reduces Problem 1 to maintaining the hint consisting of a vector q and a normalization estimate $\widetilde{Z}$. We mention that Proposition 2.5 is a strict generalization of a corresponding building block in (van Apeldoorn & Gilyén, 2019), which only used the all-ones vector as q.

Approaches for Problem 1. We now overview our improved solution to Problem 1 via efficient use of Proposition 2.5. To motivate our solution, we outline three solutions to Problem 1 offering different tradeoffs in the overall quality ρC. The first only uses classical information and does not use Proposition 2.5 at all, the second uses Proposition 2.5 but maintains no history across iterates, and the third (building upon the first two) is our approach.

Solution 1: (Grigoriadis & Khachiyan, 1995). A standard way to solve Problem 1 is to explicitly update $w = \mathbf{A}^\top x$ and $\exp(w)$, and exactly maintain the normalizing constant Z. This allows us to sample from p in $\widetilde{O}(1)$ time. Since w changes by one row of $\mathbf{A}$ under a 1-sparse Update operation to x, this is implementable in $O(n)$ time per iteration. We can view this as an instance of the scheme (1) with $q = p$, $C = 1$, and $\rho = 1$. It yields the (unbalanced) tradeoff for Problem 1 of $\mathcal{T}_{\text{samp}} = \widetilde{O}(1)$ and $\mathcal{T}_{\text{update}} = O(n)$.

Solution 2: (van Apeldoorn & Gilyén, 2019). A recent work (van Apeldoorn & Gilyén, 2019) introduced a quantum implementation of the scheme (1) with an improved tradeoff. The (van Apeldoorn & Gilyén, 2019) scheme first uniformly samples, which in the language of (1) means $q = \mathbf{1}_n$ and $\rho = n$. It then applies quantum maximum finding (Dürr & Høyer, 1996) to obtain an approximate maximum entry of w, which they show takes time $\widetilde{O}(\beta \cdot \sqrt{n})$; for the sake of simplicity here, we assume this exactly yields $w_{\max} := \max_{j \in [n]} w_j$. Finally, the acceptance probability $\frac{p_j}{Cq_j}$ is set to $\exp(w_j - w_{\max})$. For $q = \mathbf{1}_n$, this translates to $p_j \cdot \exp(w_{\max} - w_j) = \frac{\exp(w_{\max})}{Z} \leq 1$, implying $C = 1$ suffices. We note this bound on C can be tight when w is very non-uniform. Overall, the (van Apeldoorn & Gilyén, 2019) scheme's update time requires maximum finding, and its sampling time (via Proposition 2.5) requires time $\widetilde{O}(\beta \cdot \sqrt{\rho C}) = \widetilde{O}(\beta \cdot \sqrt{n})$. For $\beta = \widetilde{O}(\frac{1}{\epsilon})$ as in Algorithm 1, this yields the balanced tradeoff $\max(\mathcal{T}_{\text{samp}}, \mathcal{T}_{\text{update}}) = \widetilde{O}\left(\sqrt{n} \cdot \epsilon^{-1}\right)$. Our key insight is to improve upon this specific choice of hint in (van Apeldoorn & Gilyén, 2019), for use of Proposition 2.5.

Solution 3: this work. We design better hints for Proposition 2.5 by executing our algorithm in phases corresponding to batches of $\approx \frac{1}{\eta}$ iterations. At the start of each phase, we use the Gibbs access afforded by Proposition 2.5 to produce a suitable hint for efficiently implementing the next phase. Our execution of this strategy, parameterized by an integer $k \in [n]$, relies on the following observations.

First, during $\lceil \frac{1}{\eta} \rceil$ iterations $t \in \{\tau + s\}_{s \in [\lceil \frac{1}{\eta} \rceil]}$ (where τ starts the phase), the dynamic Gibbs distribution p_t (where t is the iteration index) changes by $O(1)$ multiplicatively, since w entrywise changes by $O(1)$ additively. Thus, the quality of a hint vector deteriorates by at most a constant in the phase, so it suffices to give a good hint at the phase start.

[5]Exactly computing Z may require time $\Omega(n)$ in standard implementations, an obstacle to runtimes $\propto \sqrt{n}$.

[6]The β in Proposition 2.5 comes from loading $\exp(w_j)$ into a quantum oracle via polynomials of degree $\approx \beta$.

Next, using access to Proposition 2.5 at the end of the previous phase, we can efficiently estimate large entries of p_τ. More precisely, we sample $\widetilde{O}(k)$ times from p_τ, and let the empirical distribution be $\tilde{q}$. Chernoff bounds show any large entry $[p_\tau]_j = \Omega(\frac{1}{k})$ will be accurately reflected in the empirical sample. Hence, we set the hint to

$$q_j = \begin{cases} \tilde{q}_j \cdot O(1) & \tilde{q}_j = \Omega(\frac{1}{k}) \\ \frac{1}{k} \cdot O(1) & \tilde{q}_j = O(\frac{1}{k}) \end{cases},$$

for appropriate constants. This yields an improved hint quality of $\rho \approx \frac{n}{k}$, since large hint entries sum to $O(1)$ (as $\tilde{q}_j \approx p_j$), and small entries sum to $O(\frac{n}{k})$. Finally, we show a similar strategy of using empirical concentration, combined with a testing variant of Proposition 2.5, yields $C = O(1)$.

This yields $\mathcal{T}_{\text{samp}} = \widetilde{O}(\beta \cdot \sqrt{n/k})$ and $\mathcal{T}_{\text{update}} = \widetilde{O}(\mathcal{T}_{\text{samp}} \cdot k\eta)$ (since we amoritize $\mathcal{T}_{\text{update}}$ over $\approx \frac{1}{\eta}$ iterations). Optimizing k for the parameter settings of Algorithm 1 gives our result. Specifically, using Theorem 2.2 for the parameter settings $T = \widetilde{O}(\epsilon^{-2})$, $\eta = \Theta(\epsilon)$, as stated in Proposition 2.1, this equates to $\max(\mathcal{T}_{\text{samp}}, \mathcal{T}_{\text{update}}) = \widetilde{O}(\sqrt{n} \cdot \epsilon^{-\frac{1}{2}})$.

3 Gibbs sampling oracle implementation

We now prove Theorem 2.2, giving our solution to Problem 1. To do so, we follow the outline in Section 2.2, and solve Problem 1 in batches of $\lceil \frac{1}{\eta} \rceil$ iterations, each called a "phase." In Section 3.1 we discuss a single phase, consisting of iterations $\tau + s$ for $s \in [\lceil \frac{1}{\eta} \rceil]$ and some initial iteration τ, assuming the invariants stated here hold at the phase start.

Invariant 1 (Approximate normalization access)**.** *We explicitly have $\widetilde{Z}_{\text{prev}}$ with $\widetilde{Z}_{\text{prev}} \in [Z_\tau, CZ_\tau]$ for some $C = O(1)$.*

Invariant 2 (Initial sampling maintenance)**.** *We have $\mathcal{O}_\tau$ solving Problem 1 in iteration τ.*

In Section 3.1 we show that assuming Invariants 1 and 2 hold at a phase start, we can perform preprocessing used to construct our hint, consisting of the estimated normalization $\widetilde{Z}$ and vector q, in an application of Proposition 2.5. This gives the cost of $\mathcal{T}_{\text{samp}}$ in Problem 1. We then show that at the phase end we can maintain Invariants 1 and 2 for use in the next phase, giving $\mathcal{T}_{\text{update}}$. In Section 3.2, we recursively call the subroutine of Section 3.1 to prove Theorem 2.2. We defer most proofs to Appendix C.

3.1 Maintaining invariants

We now show how to construct the "hint" q used throughout a phase (starting in iteration τ) given access to $\mathcal{O}_\tau$, and bound $\rho = \|q\|_1$ which quantifies the quality of our hint, assuming Invariants 1 and 2 hold. We use a multiplicative stability property of the relevant distributions within a phase, whose proof is standard and in Appendix C.

Lemma 3.1. *For all $s \in [\lceil \frac{1}{\eta} \rceil]$, $Z_{\tau+s} \in \left[\frac{1}{3} Z_\tau, 3Z_\tau\right]$ and $p_{\tau+s} \in \left[\frac{1}{9} p_\tau, 9p_\tau\right]$ entrywise.*

Our computation of the overestimating vector q is parameterized by $k \in [n]$ fixed throughout this section. We set q to an upscaled empirical distribution of $\approx k$ draws from $\mathcal{O}_\tau$.

Lemma 3.2. *Let $k \in [n]$, $\alpha \in (0, 1)$, and suppose $\delta \le \frac{1}{16k}$. Draw $N = \Theta(k \log \frac{n\eta T}{\alpha})$ samples from $\mathcal{O}_\tau$ for an appropriately large constant, and let $\tilde{q} \in \Delta^n$ be the empirical distribution over these N samples. Define $\mathcal{B} := \{i \in [n] \mid \tilde{q}_i \ge \frac{1}{2k}\}$. Then with probability $\ge 1 - \frac{\alpha}{2\lceil \eta T \rceil}$, for*

$$q_j = \begin{cases} 18\tilde{q}_j & j \in \mathcal{B} \\ \frac{18}{k} & j \notin \mathcal{B} \end{cases},$$

$\|q\|_1 = O(\frac{n}{k})$ and $q \ge p_{\tau+s}$ entrywise, for all $s \le \frac{1}{\eta}$.

The proof is based on Chernoff bounds for handling $j \in \mathcal{B}$ and $j \notin \mathcal{B}$ separately, and is deferred to Appendix C. We then obtain the following by combining the hint quality in Lemma 3.2 with Invariant 1 and Proposition 2.5.

Corollary 3.3. *Assume Invariants 1, 2 hold for the phase consisting of iterations $\tau + s$, $s \in [\lceil \frac{1}{\eta} \rceil]$. We can solve Problem 1 for the phase with probability $\ge 1 - \frac{\alpha}{2\lceil \eta T \rceil}$, and $\mathcal{T}_{\text{samp}} := O\left(\sqrt{\frac{n}{k}} \cdot T\eta \log^4 \frac{mn}{\delta}\right)$.*

We next show how to maintain Invariant 1 at iteration $\tau' := \tau + \lceil \frac{1}{\eta} \rceil$, for use in the next phase, and bound the cost; Invariant 2 follows immediately from Corollary 3.3. By combining Lemma 3.1 with Invariant 1, $Z_{\tau'} \in [\frac{\widetilde{Z}_{\text{prev}}}{3C}, 3\widetilde{Z}_{\text{prev}}]$, which suggests using $3\widetilde{Z}_{\text{prev}} = \widetilde{Z}$ for the next phase. However, this would lead to an exponential blowup in the multiplicative range C. To sidestep this, we develop a tester for a hidden parameter governing a success probability, used to give a refined estimate $\widetilde{Z}$. We require a corollary of Proposition 2.5, whose proof is in Appendix C.

Corollary 3.4. *Following notation of Proposition 2.5, let $R := \frac{\widetilde{Z}}{Z}$. There is a quantum oracle $\mathcal{O}_{\text{test}}$ which can be implemented under T Update calls to x in $O(T \log m)$ time, and has query cost $O(\sqrt{\rho C} \cdot \beta \log^4 \frac{Cmn}{\ell\delta})$. Furthermore, for explicitly known constants C_ℓ and C_u, $\mathcal{O}_{\text{test}}$ returns "success" with probability p for $\frac{C_\ell}{\sqrt{R\rho}} \le p \le \frac{C_u}{\sqrt{R\rho}}$.*

Corollary 3.4 differs from Proposition 2.5 in that it returns a Boolean-valued answer (as opposed to a sample), and has a success probability parameterized by explicit constants. We use Corollary 3.4 to maintain Invariant 1; the proof is again based on Chernoff bounds, and deferred to Appendix C.

Lemma 3.5. *Assume Invariants 1, 2 hold for iterations $\tau + s$, $s \in [\lceil \frac{1}{\eta} \rceil]$, and suppose $C \ge \frac{4C_u^2}{C_\ell^2}$ for $C = O(1)$, where C_u and C_ℓ are the constants from Corollary 3.4. Further, suppose we have obtained q satisfying the conclusion of*

Lemma 3.2 (i.e. that the algorithm in Lemma 3.2 succeeded). We can determine $\widetilde{Z}$ such that $\widetilde{Z} \in [Z_{\tau'}, CZ_{\tau'}]$ with probability $\geq 1 - \frac{\alpha}{2\lceil \eta T \rceil}$, in time $O\left(\sqrt{\frac{n}{k}} \cdot T\eta \log^4 \frac{mn}{\delta} \log \frac{\eta T}{\alpha}\right)$.

3.2 Proof of Theorem 2.2

Theorem 2.2. *Let $\alpha \in (0,1)$, $\delta \leq \eta$. Given a quantum oracle for $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $\|\mathbf{A}\|_{\max} \leq 1$, we can solve Problem 1 with probability $\geq 1 - \alpha$, and $\max(\mathcal{T}_{\text{samp}}, \mathcal{T}_{\text{update}}) = O(1 + \sqrt{n}T\eta \log^4 \frac{mn}{\delta} \cdot (\sqrt{\eta \log \frac{n\eta T}{\alpha}} + \eta \log \frac{n\eta T}{\alpha}))$, and an initialization cost of $O(\eta^3 T^3 \log^4 \frac{n\eta T}{\delta} + \log^7 \frac{n\eta T}{\delta})$.*

Proof. For any $k \in [n]$, we can solve Problem 1 with probability $\geq 1 - \alpha$ and $\mathcal{T}_{\text{samp}} = O\left(\sqrt{\frac{n}{k}} \cdot T\eta \log^4 \frac{mn}{\delta}\right)$, $\mathcal{T}_{\text{update}} = O\left(\left(\sqrt{\frac{n}{k}} \cdot T\eta \log^4 \frac{mn}{\delta}\right) \cdot k\eta \log \frac{n\eta T}{\alpha}\right)$. This follows from combining Lemma 3.2 (amoritized over $\lceil \frac{1}{\eta} \rceil$ iterations), Corollary 3.3, and Lemma 3.5, and taking a union bound over at most $\lceil \eta T \rceil$ phases. Here we note the cost of $\log m$ per iteration to support Update costs to x in Lemma 2.4, Proposition 2.5, and Corollary 3.4 is not dominant. By choosing $k = \Theta(\max(1, (\eta \log \frac{mn}{\alpha\epsilon})^{-1}))$, we balance the costs of $\mathcal{T}_{\text{samp}}$ and $\mathcal{T}_{\text{update}}$, yielding the conclusion. By picking an appropriate constant in the definition of k, we have $\delta \leq \eta \implies \delta \leq \frac{1}{16k}$ as required by Lemma 3.2, the only component specifying a bound on δ. ☐

Acknowledgments

We thank András Gilyén for communication regarding the prior work (van Apeldoorn & Gilyén, 2019), and the anonymous reviewers for their valuable comments. AB was supported in part by the DOE QuantISED grant DE-SC0020360, by the AFOSR under grant FA9550-21-1-0392, and by the U.S. DOE Office of Science under Award Number DE-SC0020266. YG was supported in part by the Stanford MS&E DE&I Research program. YJ was supported in part by a Stanford Graduate Fellowship and a Danzig-Lieberman Graduate Fellowship. AS was supported in part by a Microsoft Research Faculty Fellowship, NSF CAREER Award CCF1844855, NSF Grant CCF-1955039, a PayPal research award, and a Sloan Research Fellowship. KT thanks Ewin Tang for her expertise on quantum linear algebra and for fielding many of our questions.

References

Aaronson, S. Read the fine print. *Nature Physics*, 11(4): 291–293, 2015.

Allen-Zhu, Z. and Li, Y. Follow the compressed leader: Faster online learning of eigenvectors and faster MMWU. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 116–125. PMLR, 2017.

Alman, J. Limits on the universal method for matrix multiplication. *Theory Comput.*, 17:1–30, 2021.

Arora, S. and Kale, S. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pp. 227–236, 2007.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

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

Brassard, G., Høyer, P., Mosca, M., and Tapp, A. Quantum amplitude amplification and estimation. *Quantum Computation and Quantum Information*, 305:53–74, 2002.

Bubeck, S. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4): 231–357, 2015.

Carmon, Y., Duchi, J. C., Sidford, A., and Tian, K. A rank-1 sketch for matrix multiplicative weights. In Beygelzimer, A. and Hsu, D. (eds.), *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pp. 589–623. PMLR, 2019a.

Carmon, Y., Jin, Y., Sidford, A., and Tian, K. Variance reduction for matrix games. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11377–11388, 2019b.

Carmon, Y., Jin, Y., Sidford, A., and Tian, K. Coordinate methods for matrix games. In Irani, S. (ed.), *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pp. 283–293. IEEE, 2020.

Chakrabarti, S., Childs, A. M., Li, T., and Wu, X. Quantum algorithms and lower bounds for convex optimization. *Quantum*, 4:221, 2020.

Chia, N.-H., Gilyén, A., Li, T., Lin, H.-H., Tang, E., and Wang, C. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. In *Proceedings of the 52nd Annual ACM*

SIGACT symposium on theory of computing, pp. 387–400, 2020.

Cohen, M. B., Lee, Y. T., and Song, Z. Solving linear programs in the current matrix multiplication time. *Journal of the ACM (JACM)*, 68(1):1–39, 2021.

Dadush, D., Natura, B., and Vègh, L. A. Revisiting tardos's framework for linear programming: faster exact solutions using approximate solvers. In Irani, S. (ed.), *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pp. 931–942. IEEE, 2020.

Duan, R., Wu, H., and Zhou, R. Faster matrix multiplication via asymmetric hashing. *arXiv preprint arXiv:2210.10173*, 2023.

Dürr, C. and Høyer, P. A quantum algorithm for finding the minimum. *CoRR*, quant-ph/9607014, 1996.

Gharibian, S. and Le Gall, F. Dequantizing the quantum singular value transformation: Hardness and applications to quantum chemistry and the quantum pcp conjecture. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 19–32, 2022.

Gilyén, A., Arunachalam, S., and Wiebe, N. Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1425–1444. SIAM, 2019.

Gilyén, A., Su, Y., Low, G. H., and Wiebe, N. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In Charikar, M. and Cohen, E. (eds.), *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pp. 193–204. ACM, 2019.

Grigoriadis, M. D. and Khachiyan, L. G. A sublinear-time randomized approximation algorithm for matrix games. *Operation Research Letters*, 18(2):53–58, 1995.

Grover, L. and Rudolph, T. Creating superpositions that correspond to efficiently integrable probability distributions. *CoRR*, abs/quant-ph/0208112, 2002.

Haah, J. Product decomposition of periodic functions in quantum signal processing. *Quantum*, 3:190, 2019.

Harrow, A. W., Hassidim, A., and Lloyd, S. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

Jiang, S., Song, Z., Weinstein, O., and Zhang, H. A faster algorithm for solving general lps. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2021, 2021*, pp. 823–832, 2021.

Jordan, S. P. Fast quantum algorithm for numerical gradient estimation. *Physical review letters*, 95(5):050501, 2005.

Karmarkar, N. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pp. 302–311, 1984.

Kerenidis, I. and Prakash, A. Quantum recommendation systems. *arXiv preprint arXiv:1603.08675*, 2016.

Kerenidis, I. and Prakash, A. A quantum interior point method for lps and sdps. *ACM Transactions on Quantum Computing*, 1(1):1–32, 2020.

Lee, Y. T. and Sidford, A. Path finding methods for linear programming: Solving linear programs in o (vrank) iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 424–433. IEEE, 2014.

Lee, Y. T. and Sidford, A. Solving linear programs with sqrt (rank) linear system solves. *arXiv preprint arXiv:1910.08033*, 2019.

Li, T., Chakrabarti, S., and Wu, X. Sublinear quantum algorithms for training linear and kernel-based classifiers. In *International Conference on Machine Learning*, pp. 3815–3824. PMLR, 2019.

Li, T., Wang, C., Chakrabarti, S., and Wu, X. Sublinear classical and quantum algorithms for general matrix games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8465–8473, 2021.

Lloyd, S., Mohseni, M., and Rebentrost, P. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014.

Nemirovski, A. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

Nemirovski, A., Juditsky, A. B., Lan, G., and Shapiro, A. Robust stochastic approximation approach to stochastic programming. *SIAM J. Optim.*, 19(4):1574–1609, 2009.

Nesterov, Y. Dual extrapolation and its applications to solving variational inequalities and related problems. *Mathematical Programing*, 109(2-3):319–344, 2007.

Neumann, J. V. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.

Rebentrost, P., Mohseni, M., and Lloyd, S. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.

Renegar, J. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical programming*, 40(1):59–93, 1988.

Sachdeva, S. and Vishnoi, N. K. Faster algorithms via approximation theory. *Found. Trends Theor. Comput. Sci.*, 9(2):125–210, 2014.

Tang, E. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 217–228, 2019.

van Apeldoorn, J. and Gilyén, A. Quantum algorithms for zero-sum games. *CoRR*, abs/1904.03180, 2019.

van Apeldoorn, J., Gilyén, A., Gribling, S., and de Wolf, R. Convex optimization using quantum oracles. *Quantum*, 4:220, 2020.

Van Apeldoorn, J., Gilyén, A., Gribling, S., and de Wolf, R. Quantum sdp-solvers: Better upper and lower bounds. *Quantum*, 4:230, 2020.

van den Brand, J. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Thirty-first Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, 2020*, pp. 259–278, 2020.

van den Brand, J., Lee, Y. T., Sidford, A., and Song, Z. Solving tall dense linear programs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 775–788, 2020.

van den Brand, J., Lee, Y. T., Liu, Y. P., Saranurak, T., Sidford, A., Song, Z., and Wang, D. Minimum cost flows, mdps, and ℓ_1-regression in nearly linear time for dense instances. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2021, 2021*, pp. 859–869, 2021.

A Solving matrix games with a Gibbs sampling oracle

In this section, we prove Proposition 2.1, which shows how to solve a zero-sum matrix game using an approximate Gibbs sampling oracle (via Algorithm 1). To briefly motivate the algorithm we use and our proof of its guarantees, we recall the problem we consider is of the form

$$\min_{v\in\Delta^n}\max_{u\in\Delta^m} f(u,v) := u^\top \mathbf{A} v, \quad \text{where} \quad \|\mathbf{A}\|_{\max} \le 1, \tag{2}$$

and we define the associated gradient operator as

$$g(u,v) = (-\mathbf{A}v, \mathbf{A}^\top u). \tag{3}$$

Taking (stochastic) mirror descent steps on the gradient operator in (2) is well-known to yield an approximate NE to the matrix game (Bubeck, 2015). We show that an approximate implementation of this strategy, combined with appropriate subsampling, efficiently yields an approximate NE. We begin by making the following observation.

Lemma A.1. *Let $u, \tilde{u} \in \Delta^m$ have $\|u-\tilde{u}\|_1 \le \delta$. Let $\tilde{g} := \mathbf{A}_{i:}$ where $i \sim \tilde{u}$, and $g := \mathbf{A}^\top u$. Then, $\|g - \mathbb{E}\tilde{g}\|_\infty \le \delta$.*

Proof. Note that $\mathbb{E}\tilde{g} = \mathbf{A}^\top \tilde{u}$, and $\left\|\mathbf{A}^\top(u-\tilde{u})\right\|_\infty \le \|u-\tilde{u}\|_1 \le \delta$ since $\|\mathbf{A}\|_{\max} \le 1$. □

We next present a variant of the classical mirror descent analysis, which bounds the expected approximation quality of iterates of Algorithm 1 prior to subsampling.

Proposition A.2. *Let $\delta \le \frac{\epsilon}{20}$, $\eta = \frac{\epsilon}{15}$ and $T \ge \frac{6\log(mn)}{\eta\epsilon}$ in Algorithm 1. Let the iterates of Algorithm 1 be $\{x_t, y_t\}_{t=0}^{T-1}$, and denote $u_t := \frac{\exp(\mathbf{A}y_t)}{\|\exp(\mathbf{A}y_t)\|_1}$, $v_t := \frac{\exp(-\mathbf{A}^\top x_t)}{\|\exp(-\mathbf{A}^\top x_t)\|_1}$ for all $0 \le t < T$. For $(\bar{u}, \bar{v}) := \frac{1}{T}\sum_{t=0}^{T-1}(u_t, v_t)$, we have*

$$\mathbb{E}\left[\max_{u\in\Delta^m} u^\top \mathbf{A}\bar{v} - \min_{v\in\Delta^n} \bar{u}^\top \mathbf{A} v\right] \le \epsilon. \tag{4}$$

Proof. By definition of the updates, at every iteration $0 \le t \le T-1$, we have

$$u_{t+1} = \text{argmin}_{u\in\Delta^m}\left\{\eta\langle -\mathbf{A}_{:j_t}, u\rangle + \sum_{i\in[m]} [u]_i \log\frac{[u]_i}{[u_t]_i}\right\},$$
$$v_{t+1} = \text{argmin}_{v\in\Delta^n}\left\{\eta\langle \mathbf{A}_{i_t:}, v\rangle + \sum_{j\in[n]} [v]_j \log\frac{[v]_j}{[v_t]_j}\right\}.$$

Consequently, by the optimality conditions of u_{t+1} and v_{t+1} respectively, we have for any $u \in \Delta^m$, $v \in \Delta^n$, and letting $V_x(x') := \sum_k [x']_k \log\frac{[x']_k}{[x]_k}$ be the KL divergence between simplex variables of appropriate dimension,

$$\begin{aligned}
\langle -\mathbf{A}_{:j}, u_t - u\rangle + \langle \mathbf{A}_{i:}, v_t - v\rangle &\le \frac{1}{\eta}\left(V_{u_t}(u) - V_{u_{t+1}}(u) + V_{v_t}(v) - V_{v_{t+1}}(v)\right) \\
&+ \left(\langle -\mathbf{A}_{:j}, u_t - u_{t+1}\rangle - \frac{1}{\eta}V_{u_t}(u_{t+1})\right) \\
&+ \left(\langle \mathbf{A}_{i:}, v_t - v_{t+1}\rangle - \frac{1}{\eta}V_{v_t}(v_{t+1})\right) \\
&\le \frac{1}{\eta}\left(V_{u_t}(u) - V_{u_{t+1}}(u) + V_{v_t}(v) - V_{v_{t+1}}(v)\right) \\
&+ \frac{\eta}{2}\|\mathbf{A}_{:j}\|_\infty^2 + \frac{\eta}{2}\|\mathbf{A}_{i:}\|_\infty^2,
\end{aligned} \tag{5}$$

where for the last inequality we use Hölder's inequality and the fact that V is 1-strongly convex in the ℓ_1 norm (by Pinsker's inequality). Averaging the above for $0 \le t < T$, and denoting $w_t := (u_t, v_t)$ and $\tilde{g}_t := (-\mathbf{A}_{:j_t}, \mathbf{A}_{i_t:})$, we obtain for any $w = (u,v) \in \Delta^m \times \Delta^n$,

$$\frac{1}{T}\sum_{t=0}^{T-1}\langle \tilde{g}_t, w_t - w\rangle \le \frac{1}{\eta T}\left(V_{u_0}(u) + V_{v_0}(v)\right) + \eta. \tag{6}$$

In the above, we further recalled the bound $\|\mathbf{A}\|_{\max} \le 1$ by assumption. In order to bound the deviation of the left-hand side from its expectation, we use a "ghost iterate" argument following (Nemirovski et al., 2009; Carmon et al., 2019b). In particular, we define iterates $\tilde{u}_t$, $\tilde{v}_t$ as follows: let $\tilde{u}_0 \leftarrow u_0$, $\tilde{v}_0 \leftarrow v_0$, and then for each $0 \le t < T$, define

$$\tilde{u}_{t+1} := \text{argmin}_{u \in \Delta^m} \left\{ \eta \langle -\mathbf{A}v_t + \mathbf{A}_{:j_t}, \bar{u} \rangle + \sum_{i \in [m]} [u]_i \log \frac{[u]_i}{[\tilde{u}_t]_i} \right\},$$
$$\tilde{v}_{t+1} := \text{argmin}_{v \in \Delta^n} \left\{ \eta \langle \mathbf{A}^\top u_t - \mathbf{A}_{:i_t}, \bar{v} \rangle + \sum_{j \in [n]} [v]_j \log \frac{[v]_j}{[\tilde{v}_t]_j} \right\},$$

where i, j above are the same coordinates as were used in defining the updates to u_{t+1} and v_{t+1}. By an analogous bound to (5), where we note $\left\|\mathbf{A}_{:j_t} - \mathbf{A}^\top v_t\right\|_\infty, \|\mathbf{A}u_t - \mathbf{A}_{i_t:}\|_\infty \le 2$,

$$\left\langle -\mathbf{A}^\top v_t + \mathbf{A}_{:j_t}, \tilde{u}_t - u \right\rangle + \langle \mathbf{A}u_t - \mathbf{A}_{i_t:}, \tilde{v}_t - v \rangle \le \frac{1}{\eta} \left(V_{\tilde{u}_t}(u) - V_{\tilde{u}_{t+1}}(u) + V_{\tilde{v}_t}(v) - V_{\tilde{v}_{t+1}}(v) \right) + 4\eta.$$

Averaging the above for $0 \le t < T$, and denoting $\tilde{w}_t := (\tilde{u}_t, \tilde{v}_t)$ and $g_t := g(w_t)$ (see (2)), we obtain for any $w = (u, v) \in \Delta^m \times \Delta^n$,

$$\frac{1}{T} \sum_{t \in [T]-1} \langle g_t - \tilde{g}_t, \tilde{w}_t - w \rangle \le \frac{1}{\eta T} \left(V_{u_0}(u) + V_{v_0}(v) \right) + 4\eta. \tag{7}$$

Summing inequalities (6) and (7), and maximizing over $w = (u, v) \in \Delta^m \times \Delta^n$, we have

$$\max_{w \in \Delta^m \times \Delta^n} \frac{1}{T} \sum_{t=0}^{T-1} \langle g_t, w_t - w \rangle \le \max_{u \in \Delta^n, v \in \Delta^m} \frac{2}{\eta T} \left(V_{u_0}(u) + V_{v_0}(v) \right) + 5\eta + \frac{1}{T} \sum_{t=0}^{T-1} \langle g_t - \tilde{g}_t, w_t - \tilde{w}_t \rangle. \tag{8}$$

Taking expectations over the above, we have

$$\begin{aligned} \mathbb{E}\left[\max_{w \in \Delta^m \times \Delta^n} \frac{1}{T} \sum_{t=0}^{T-1} \langle g_t, w_t - w \rangle \right] &\le \max_{u \in \Delta^n, v \in \Delta^m} \frac{2}{\eta T} \left[V_{u_0}(u) + V_{v_0}(v) \right] \\ &\quad + 5\eta + \mathbb{E}\left[\frac{1}{T} \sum_{t=0}^{T-1} \langle g_t - \tilde{g}_t, w_t - \tilde{w}_t \rangle \right] \\ &\overset{(i)}{\le} \frac{2\log(mn)}{\eta T} + 5\eta + \frac{1}{T} \sum_{t \in [T]-1} \langle g_t - \mathbb{E}\tilde{g}_t, w_t - \bar{w}_t \rangle, \\ &\overset{(ii)}{\le} \frac{2\log(mn)}{\eta T} + 5\eta + 4\delta \overset{(iii)}{\le} \epsilon. \end{aligned}$$

In the above, (i) used the diameter bound of the KL divergence from the uniform distribution, i.e. $\max_{u \in \Delta^m} V_{u_0}(u) = \log m$ (and a similar bound for $V_{v_0}(v)$). Further, (ii) uses that $\tilde{g}_t$ is conditionally independent of w_t and $\tilde{w}_t$, and by the assumption on the Gibbs sampler $\|g_t - \mathbb{E}\tilde{g}_t\|_\infty \le \delta$ (via Lemma A.1), and Hölder, and (iii) uses our choices of T, η and δ.

Finally, we note that the desired claim follows by linearity: for any $w = (u, v)$,

$$\begin{aligned} \frac{1}{T} \sum_{t=0}^{T-1} \langle g_t, w_t - w \rangle &= \left\langle g\left(\frac{1}{T} \sum_{t=0}^{T-1} w_t \right), \frac{1}{T} \sum_{t=0}^{T-1} w_t - w \right\rangle \\ &= u^\top \mathbf{A}\bar{v} - \bar{u}^\top \mathbf{A}v. \end{aligned}$$

□

By using a simple martingale argument (inspired by those in (Allen-Zhu & Li, 2017; Carmon et al., 2019a)) to bound the error term in (8), we show that the guarantee of Proposition A.2 holds with high probability.

Corollary A.3. *Let $\alpha \in (0,1)$, and let $\delta \le \frac{\epsilon}{20}$, $\eta = \frac{\epsilon}{20}$ and $T \ge \frac{8\log(mn)}{\eta\epsilon} + \frac{2048\log\frac{1}{\alpha}}{\epsilon^2}$ in Algorithm 1. Then with probability at least $1-\alpha$, following notation of Proposition A.2, $(\bar{u}, \bar{v})$ are an ϵ-approximate NE for $\mathbf{A}$.*

Proof. Consider the filtration given by $\mathcal{F}_t = \sigma(u_0, v_0, \tilde{g}_0, \cdots, \tilde{g}_t, u_{t+1}, v_{t+1})$. We will bound the terms $\sum_{t=0}^{T-1} \langle g_t - \tilde{g}_t, w_t - \bar{w}_t\rangle$ in (4). To do so, we define a martingale difference sequence of the form $D_t := \langle g_t - \tilde{g}_t, w_t - \bar{w}_t\rangle - \langle g_t - \mathbb{E}\left[\tilde{g}_t | \mathcal{F}_{t-1}\right], w_t - \bar{w}_t\rangle$ which is adapted to the filtration $\mathcal{F}_t$. We first note that $D_{t-1} \le \left\|g_{t-1} - \tilde{g}_{t-1}\right\|_\infty \left\|w_{t-1} - \bar{w}_{t-1}\right\|_1 \le 8$ with probability 1. Consequently, applying the Azuma-Hoeffding inequality yields

$$\sum_{t=0}^{T-1} D_t \le \sqrt{128T\log\frac{1}{\alpha}} \text{ with probability} \ge 1-\alpha.$$

Plugging this back into (8) and using the KL divergence range bound, Lemma A.1 with our definition of $\mathcal{O}^{\text{gibbs}}$, and choices of parameters, we thus have with probability $1-\alpha$,

$$\max_{w \in \Delta^m \times \Delta^n} \frac{1}{T}\sum_{t=0}^{T-1} \langle g_t, w_t - w\rangle \le \frac{2\log mn}{\eta T} + 5\eta + 4\delta + \sqrt{\frac{128\log\frac{1}{\alpha}}{T}} \le \epsilon. \tag{9}$$

The remainder of the proof follows analogously to Proposition A.2. □

The Gibbs sampling oracles implicitly maintain access to $u_t \propto \exp(\mathbf{A}y_t)$ and $v_t \propto \exp(-\mathbf{A}^\top x_t)$, which by averaging gives $(\bar{u}, \bar{v}) = \frac{1}{T}\sum_{t=0}^{T-1}(u_t, v_t)$ as one approximate equilibrium as guaranteed in Corollary A.3. To turn the implicitly maintained iterates into an actual classic output, we subsample the iterates. Below we formally show one can take the empirical average of independent samples from distributions close to $\bar{u}$ and $\bar{v}$ to also obtain an approximate equilibrium (with the same approximation factor up to constant factors) with high probability.

Lemma A.4. *Suppose $\bar{u} = \frac{1}{T}\sum_{t=0}^{T-1} u_t$ for $\{u_t\}_{t=0}^{T-1} \subset \Delta^m$ and $\bar{v} = \frac{1}{T}\sum_{t=0}^{T-1} v_t$ for $\{v_t\}_{t=0}^{T-1} \subset \Delta^n$ are an ϵ-approximate NE for $\mathbf{A}$. Further suppose that for some $\delta \in (0,1)$, $\{\tilde{u}_t\}_{t=0}^{T-1} \subset \Delta^m$, $\{\tilde{v}_t\}_{t=0}^{T-1} \subset \Delta^n$, and for all $0 \le t < T-1$, we have $\left\|\tilde{u}_t - u_t\right\|_1 \le \delta$ and $\left\|\tilde{v}_t - v_t\right\|_1 \le \delta$. Let $\hat{u} = \frac{1}{T}\sum_{t=0}^{T-1} e_{i_t}$ where each $e_{i_t} \in \mathbb{R}^m$ is sampled independently according to $\tilde{u}_t$; similarly, let $\hat{v} = \frac{1}{T}\sum_{t=0}^{T-1} e_{j_t}$ where each $e_{j_t} \in \mathbb{R}^n$ is sampled independently according to $\tilde{v}_t$. Suppose $T \ge \frac{16\log\frac{mn}{\alpha}}{\epsilon^2}$. Then with probability at least $1-\alpha$, $(\hat{u}, \hat{v})$ are a $(2\epsilon + 2\delta)$-approximate NE for $\mathbf{A}$.*

Proof. First, let $\tilde{u}_{\text{avg}} = \frac{1}{T}\sum_{t=0}^{T-1} \tilde{u}_t$ and $\tilde{v}_{\text{avg}} = \frac{1}{T}\sum_{t=0}^{T-1} \tilde{v}_t$. By convexity of norms, we have $\left\|\tilde{u}_{\text{avg}} - \bar{u}\right\|_1 \le \delta$ and $\left\|\tilde{v}_{\text{avg}} - \bar{v}\right\|_1 \le \delta$, and hence under the NE approximation guarantee of $(\bar{u}, \bar{v})$ and Hölder's inequality,

$$\max_{u \in \Delta^m} u^\top \mathbf{A}\tilde{v}_{\text{avg}} - \min_{v \in \Delta^m} \tilde{u}_{\text{avg}}^\top \mathbf{A} v \le \epsilon + 2\delta.$$

Let z be a fixed vector in $[-1,1]^n$. By Hoeffding's inequality, since each random variable $\langle z, e_{j_t}\rangle$ lies in the range $[-1,1]$ and $\mathbb{E}\hat{v} = \tilde{v}_{\text{avg}}$, we have that

$$\Pr\left[\left|\langle z, \hat{v} - \tilde{v}_{\text{avg}}\rangle\right| \ge \frac{\epsilon}{2}\right] \le 2\exp\left(-\frac{T\epsilon^2}{8}\right) \le \frac{\alpha}{m+n}. \tag{10}$$

Next, note that $\max_{u \in \Delta^m} u^\top \mathbf{A}\tilde{v}_{\text{avg}}$ is achieved by a basis vector $u = e_i$. Hence, applying a union bound over (10) for all $z = \mathbf{A}_{i:}$ shows that with probability at least $1 - \frac{\alpha m}{m+n}$,

$$\max_{u \in \Delta^m} u^\top \mathbf{A}\hat{v} \le \max_{u \in \Delta^m} u^\top \mathbf{A}\tilde{v}_{\text{avg}} + \frac{\epsilon}{2}.$$

By symmetry, with probability at least $1 - \frac{\alpha n}{m+n}$,

$$\min_{v \in \Delta^n} \hat{u}^\top \mathbf{A} v \ge \min_{v \in \Delta^n} \tilde{u}_{\text{avg}}^\top \mathbf{A} v - \frac{\epsilon}{2}.$$

The conclusion follows from a union bound, and combining the above three displays. □

Finally, we put these pieces together to give a complete guarantee.

Proposition 2.1. *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ satisfy $\|\mathbf{A}\|_{\max} \le 1$ and $\epsilon, \alpha \in (0, 1)$. Let $\delta \le \frac{\epsilon}{20}$, $\eta = \frac{\epsilon}{60}$, and $T = \Theta(\epsilon^{-2} \log \frac{mn}{\alpha})$ for an appropriate constant. With probability $\ge 1 - \alpha$, Algorithm 1 outputs an ϵ-approximate NE for $\mathbf{A}$.*

Proof. We follow notation of Proposition A.2. By applying Corollary A.3 (up to constant factors), we have that with probability at least $1 - \frac{\alpha}{2}$, $\bar{u} := \frac{1}{T} \sum_{t=0}^{T-1} u_t$ and $\bar{v} := \frac{1}{T} \sum_{t=0}^{T-1} v_t$ satisfy

$$\max_{u \in \Delta^m} u^\top \mathbf{A} \bar{v} - \min_{v \in \Delta^n} \bar{u}^\top \mathbf{A} v \le \frac{\epsilon}{3}.$$

Finally, Lemma A.4 (with failure probability $\frac{\alpha}{2}$) and a union bound yields the desired conclusion. □

B Quantum rejection sampling with a hint

In this section, we prove Proposition 2.5, which gives a dynamic quantum rejection sampling subroutine and bounds its cost of implementation. Our result is an extension of analogous developments in (van Apeldoorn & Gilyén, 2019), but are stated more generally to allow for the use of an appropriate "hint" vector in the rejection sampling procedure. We build up to our main result in several pieces.

Amplitude amplification. First, for a quantum decision algorithm which applies unitary $\mathbf{U}$ and then measures, yielding an accepting state with probability α, quantum amplification (Brassard et al., 2002) shows we can apply $\mathbf{U} \approx \alpha^{-\frac{1}{2}}$ times to obtain an accepting state with high probability.

Proposition B.1 (Theorem 3, (Brassard et al., 2002)). *Let $S \subseteq \{0, 1\}^s$, let $\mathbf{U}$ be a s-qubit quantum oracle, and let α be the probability that measuring the result of applying $\mathbf{U}$ yields an accepting state. There is a (quantum) algorithm using $O(\alpha^{-\frac{1}{2}} \log \frac{1}{\delta})$ queries to $\mathbf{U}$ and $O(\log s \log \frac{1}{\delta})$ additional time that returns s with $s \in S$ with probability $\ge 1 - \delta$.*

Loading from trees. Given a dynamic vector $x \in \mathbb{R}^m_{\ge 0}$ which is supported in an appropriate efficient data structure $\mathsf{SamplerTree}$ (see Lemma 2.4), and a known bound $\beta \ge \|x\|_1$, we recall a result of (Grover & Rudolph, 2002) which allows us to form a superposition of the entries in x (suitably rescaled).

Lemma B.2. *Let $x \in \mathbb{R}^m_{\ge 0}$ correspond to an instance of $\mathsf{SamplerTree}$, and $\beta \ge \|x\|_1$. We can maintain a quantum oracle $\mathcal{O}_{\mathsf{SamplerTree}}$ which takes $O(\log m)$ time to apply, such that the total cost of building $\mathcal{O}_{\mathsf{SamplerTree}}$ after T calls to Update is $O(T \log m)$, and*

$$\mathcal{O}_{\mathsf{SamplerTree}} |0\rangle^{\otimes(a+1)} = \sum_{i \in [m]} \sqrt{\frac{x_i}{\beta}} |0\rangle|i\rangle + \sqrt{1 - \frac{\|x\|_1}{\beta}} |1\rangle|g\rangle.$$

Proof. This is implicit in (Grover & Rudolph, 2002). We first apply a 1-qubit gate to condition on selecting from the tree (with probability $\frac{\|x\|_1}{\beta}$), and then apply the (Grover & Rudolph, 2002) procedure conditioned on the first qubit being $|0\rangle$, which controls for one qubit at a time while propagating subtree sums (provided by $\mathsf{SamplerTree}$ via $\mathsf{SubtreeSum}$). The cost to build the circuit follows because on an Update we need to change the gates corresponding to the relevant leaf-to-root path.

□

Corollary B.3. *Let $x \in \mathbb{R}^m_{\ge 0}$ correspond to an instance of $\mathsf{SamplerTree}$, and let $\beta \ge \|x\|_1$, and suppose $\mathbf{A} \in \mathbb{R}^{m \times n}$ has $\|\mathbf{A}\|_{\max} \le 1$. We can maintain a quantum oracle $\mathcal{O}_{\mathbf{A}^\top x}$ which takes $O(\log m)$ time to apply, with total building cost $O(T \log m)$ after T calls to Update, such that for any $j \in [n]$,*

$$\mathcal{O}_{\mathbf{A}^\top x} |0\rangle^{\otimes(a+2)} |j\rangle = |0\rangle \left(\sum_{i \in [m]} \sqrt{\frac{\mathbf{A}_{ij} x_i}{\beta}} |0\rangle|i\rangle + |1\rangle|g\rangle \right) |j\rangle.$$

Proof. We apply $\mathcal{O}'_{\mathbf{A}}$ (see Section 1.4) to the output of $\mathcal{O}_{\mathsf{SamplerTree}}$, ignoring the additional qubit. □

We remark here that the additional qubit in Corollary B.3 will shortly become useful in constructing an appropriate block-encoding of a scaling of **diag** $\left(\mathbf{A}^\top x\right)$.

Polynomial approximation. In order to give approximate Gibbs samplers for the types of dynamic vectors Algorithm 1 encounters, we further require some tools from polynomial approximation theory. We first state a helper result on boundedly approximating the exponential, a variant of which was also used in (van Apeldoorn & Gilyén, 2019). We provide a proof in Appendix D.

Lemma B.4 (Lemma 7, (van Apeldoorn & Gilyén, 2019))**.** *Let $\beta \geq 1$, $\xi \leq \frac{1}{10}$. There is a polynomial $\mathcal{P}_{\beta,\xi}$ of degree $O(\beta \log \frac{1}{\xi})$ such that* $\max_{x\in[-1,1]} |\mathcal{P}_{\beta,\xi}(x)| \leq 3$ *and* $\max_{x\in[-1,0]} |\mathcal{P}_{\beta,\xi}(x) - \exp(\beta x)| \leq \xi$.

Next, we state a further corollary of Lemma B.4 to be used in our rejection sampler.

Corollary B.5. *Let $B, \delta \geq 0$ and suppose $v \in \mathbb{R}^n$ has $\|v\|_\infty \leq B$. Further, suppose for some $c \geq 0$, $-c \leq \max_{j\in[n]} v_j \leq 0$. Let $q \in \mathbb{R}^n_{\geq 0}$ satisfy $q_j \in [\ell, 1]$ entrywise. Finally, define $u_j := \frac{v_j}{2B}$ entrywise. There is a degree-Δ polynomial $\mathcal{P}$, for $\Delta = O(B \cdot (c + \log \frac{n}{\ell\delta}))$, such that for $w_j := \mathcal{P}(u_j)^2 q_j$ and $z_j := \exp(2Bu_j)q_j$ entrywise,*

$$\left\| \frac{w}{\|w\|_1} - \frac{z}{\|z\|_1} \right\|_1 \leq \delta. \tag{11}$$

Moreover, $\max_{x\in[-1,1]} |\mathcal{P}(x)| \leq \frac{1}{2}$, and $\|w\|_1 \geq \frac{1-\delta}{36}\|z\|_1$.

Proof. Assume $\delta \leq 2$ else the statement is clearly true. First, $u_j \in [-\frac{1}{2}, 0]$ entrywise by the stated assumptions (since $v_j \in [-B, 0]$ entrywise). Let $\mathcal{P}_{\beta,\xi}(\cdot)$ be the polynomial given by Lemma B.4 which ξ-approximates $\exp(\beta\cdot)$ on $[-\frac{1}{2}, 0]$. We define

$$\mathcal{P}(u) := \frac{1}{6}\mathcal{P}_{B,\xi}(u), \text{ for } \xi := \frac{\delta\ell}{6n\exp(c)}.$$

The degree bound and absolute value bound of this polynomial follows immediately from Lemma B.4, so it remains to show the distance bound. The guarantees of Lemma B.4 then imply for all $j \in [n]$,

$$|6\mathcal{P}(u_j) - \exp(Bu_j)| \leq \xi. \tag{12}$$

We further have that $u_j \leq 0$, so $\exp(Bu_j) \leq 1$. Hence, we also have

$$|6\mathcal{P}(u_j) + \exp(Bu_j)| \leq 2 + \xi \leq 3.$$

Combining yields for all $j \in [n]$,

$$\left|36\mathcal{P}(u_j)^2 - \exp(2Bu_j)\right| \leq 3\xi. \tag{13}$$

Next, let $y_j := 36w_j$ for all $j \in [n]$, and note that $\frac{y}{\|y\|_1} = \frac{w}{\|w\|_1}$. We bound

$$\begin{aligned}\left\| \frac{w}{\|w\|_1} - \frac{z}{\|z\|_1} \right\|_1 &= \sum_{j\in[n]} \left| \frac{y_j}{\|y\|_1} - \frac{z_j}{\|z\|_1} \right| \leq \sum_{j\in[n]} \left| \frac{y_j}{\|y\|_1} - \frac{y_j}{\|z\|_1} \right| + \sum_{j\in[n]} \left| \frac{y_j}{\|z\|_1} - \frac{z_j}{\|z\|_1} \right| \\ &\leq \left| 1 - \frac{\|y\|_1}{\|z\|_1} \right| + \frac{\|y - z\|_1}{\|z\|_1} \leq \frac{2\|y - z\|_1}{\|z\|_1}.\end{aligned} \tag{14}$$

By using the definitions of y, z and (13), as well as the assumed ranges on q,

$$\|y - z\|_1 \leq 3n\xi,\ \|z\|_1 \geq \ell\exp(-c).$$

The second inequality used that some $v_j = 2Bu_j$ is at least $-c$ by assumption. Combining the above display with (14) and the definition of ξ concludes the proof of (11). Finally, using the bounds on $\|y - z\|_1$, $\|z\|_1$ above shows that

$$\|w\|_1 = \frac{1}{36}\|y\|_1 \geq \frac{1-\delta}{36}\|z\|_1.$$

□

Block-encoding. Our approximate Gibbs oracle follows an implementation strategy pioneered by (Gilyén et al., 2019) termed "block-encoding." Specifically, we follow (Gilyén et al., 2019) and say that $\mathbf{U}$, an $(a+\ell)$-qubit quantum gate, is an ℓ-bit block-encoding of $\mathbf{M}$ if the top-left $2^a \times 2^a$ submatrix of $\mathbf{U}$ is $\mathbf{M}$. Block-encoded matrices admit efficient composable operations, such as the application of linear combinations and bounded polynomials. We summarize these properties in the following.

Proposition B.6 (Lemma 52, (Gilyén et al., 2019))**.** *Let $\mathbf{U}_1$ and $\mathbf{U}_2$ be ℓ-bit block-encodings of $\mathbf{M}_1$, $\mathbf{M}_2$ of the same size. There is an $O(\ell)$-bit block-encoding of $\frac{1}{2}\mathbf{M}_1 + \frac{1}{2}\mathbf{M}_2$ which takes the same asymptotic time to apply as applying $\mathbf{U}_1$ and $\mathbf{U}_2$.*

Proposition B.7 (Theorem 56, (Gilyén et al., 2019))**.** *Let $\mathbf{U}$ be an ℓ-bit block-encoding of $\mathbf{M}$, and $\mathcal{P} : [-1,1] \to [-\frac{1}{2}, \frac{1}{2}]$ be a degree-Δ polynomial. There is an $O(\ell)$-bit block-encoding of $\mathcal{P}(\mathbf{M})$ which can be applied in $O(\Delta)$ applications of $\mathbf{U}$ and $\mathbf{U}^\dagger$ and $O(\ell\Delta)$ additional time.*

We also demonstrate that an application of Corollary B.3 yields a simple block-encoding of $\mathbf{diag}\left(\frac{\mathbf{A}^\top x}{\beta}\right)$. A similar construction previously appeared in (van Apeldoorn & Gilyén, 2019).

Corollary B.8. *Let $x \in \mathbb{R}^m_{\geq 0}$ correspond to an instance of* SamplerTree*, and $\beta \geq \|x\|_1$. Let $\mathbf{M} := \mathbf{diag}\left(\frac{\mathbf{A}^\top x}{\beta}\right)$ and $\mathbf{U} := \mathcal{O}^*_{\mathbf{A}^\top x}(\mathrm{SWAP}_{12} \otimes \mathbf{I})\mathcal{O}_{\mathbf{A}^\top x}$, where SWAP_{12} swaps the first two qubits and $\mathcal{O}_{\mathbf{A}^\top x}$ is from Corollary B.3. Then $\mathbf{U}$ is a block-encoding of $\mathbf{M}$, and can be applied in time $O(\log m)$, with total building cost $O(T \log m)$ after T calls to* Update*.*

Proof. Define $w_{ij} := \frac{\mathbf{A}_{ij} x_i}{\beta}$ for convenience. By the definition of $\mathcal{O}_{\mathbf{A}^\top x}$, we have that

$$(\mathrm{SWAP}_{12} \otimes \mathbf{I})\, \mathcal{O}_{\mathbf{A}^\top x}\left(|0\rangle^{\otimes(a+2)}|j\rangle\right) = \left(|00\rangle \sum_{i\in[m]} \sqrt{w_{ij}}|i\rangle + |10\rangle|g\rangle\right)|j\rangle.$$

Hence, for $j, j' \in [n]$, we compute $\langle j'|\langle 0|^{\otimes(a+2)}\mathbf{U}|0\rangle^{\otimes(a+2)}|j\rangle$ as:

$$\langle j'| \left(|00\rangle \sum_{i\in[m]} \sqrt{w_{ij}}|i\rangle + |01\rangle|g\rangle\right)^* \left(|00\rangle \sum_{i\in[m]} \sqrt{w_{ij}}|i\rangle + |10\rangle|g\rangle\right)|j\rangle$$
$$= \begin{cases} \sum_{i\in[m]} w_{ij} = \frac{[\mathbf{A}^\top x]_j}{\beta} & j = j' \\ 0 & j \neq j' \end{cases}.$$

In particular the $|01\rangle$ and $|10\rangle$ terms disappear, and $|j\rangle$, $|j'\rangle$ are orthogonal unless $j = j'$. In the above, we required that $\sqrt{w_{ij}}^*\sqrt{w_{ij}} = w_{ij}$, which is only true if w_{ij} is nonnegative. To bypass this issue, we will implement the two copies of $\mathcal{O}_{\mathbf{A}^\top x}$ in slightly different ways, to obtain the correct signing. For notational clarity, we let $\mathcal{O}^L$ be the oracle which is conjugated on the left and $\mathcal{O}^R$ be the oracle on the right, such that $\mathbf{U} = (\mathcal{O}^L)^*(\mathrm{SWAP}_{12} \otimes \mathbf{I})(\mathcal{O}^R)$. Note that x is entrywise nonnegative and $\beta > 0$, and hence the only factor determining the sign of w_{ij} is $\mathbf{A}_{ij}$. When $\mathbf{A}_{ij} \geq 0$, we will define the oracles $\mathcal{O}'_{\mathbf{A}}$ used to load $\sqrt{\mathbf{A}_{ij}}$ for $\mathcal{O}^L$ and $\mathcal{O}^R$ in a consistent way (i.e. use the same-signed square root), so that $\sqrt{w_{ij}}^2 = w_{ij}$. When $\mathbf{A}_{ij} < 0$ we will define them in an inconsistent way, so that after the conjugation operation, $-\sqrt{w_{ij}}\sqrt{w_{ij}} = w_{ij}$. We have thus shown that $\langle 0|^{\otimes(a+2)}\mathbf{U}|0\rangle^{\otimes(a+2)} = \mathbf{M}$ which implies the first conclusion. To see the second, all our gates are reversible (arithmetic circuits are reversible, and $\mathcal{O}_{\mathbf{A}}$ is its own inverse), and hence the complexity of applying $\mathcal{O}^*_{\mathbf{A}^\top x}$ is the same as $\mathcal{O}_{\mathbf{A}^\top x}$. $\square$

Finally, we put together the pieces and prove Proposition 2.5, which we use repeatedly throughout the paper to implement our Gibbs sampling oracles.

Proposition 2.5. *Let $x \in \mathbb{R}^m_{\geq 0}$ correspond to an instance of* SamplerTree*, and $\beta \geq \|x\|_1$. Let p be the Gibbs distribution associated with $\mathbf{A}^\top x$, let $Z := \sum_{j\in[n]} \exp([\mathbf{A}^\top x]_j)$ and $\widetilde{Z} \in [Z, CZ]$ for some $C \geq 1$. Finally, let $q \in \mathbb{R}^n$ have entries classically queriable in $O(1)$ time, satisfy $q \geq p$ entrywise, $q_j \in [\frac{\delta}{n}, 1]$ for all $j \in [n]$, and $\|q\|_1 = \rho$. Suppose $\widetilde{Z}$, C, ρ, and β are explicitly known. Given a quantum oracle for $\mathbf{A} \in \mathbb{R}^{m\times n}$ (defined in Section 1.4) with $\|\mathbf{A}\|_{\max} \leq 1$, we can implement a δ-approximate Gibbs oracle which has query cost $O(\sqrt{\rho C} \cdot \beta \log^4\left(\frac{Cmn}{\delta}\right))$. The total additional cost incurred if x undergoes T* Update *calls which preserve the invariants on $\widetilde{Z}, C, \rho, \beta$ is $O(T \log m)$.*

Proof. Throughout the proof, let $\delta \leftarrow \min(\frac{1}{2}, \delta)$ and $B := 4(\beta + \log(\frac{Cn}{\delta}))$. Also define $\ell := \frac{\delta}{n}$ (following notation of Corollary B.5). We first observe that since $\max_{j\in[n]}[\mathbf{A}^\top x]_j \le \log Z \le \max_{j\in[n]}[\mathbf{A}^\top x]_j + \log n$,

$$-\log(Cn) \le \max_{j\in[n]}[\mathbf{A}^\top x]_j - \log\left(\widetilde{Z}q_j\right) \le 0.$$

Here, the upper bound used that for all $j \in [n]$, $\exp([\mathbf{A}^\top x]_j - \widetilde{Z}q_j) = \frac{p_j}{q_j} \cdot \frac{Z}{\widetilde{Z}} \le 1$ by assumption. Hence, for $v := \mathbf{A}^\top x - \log(\widetilde{Z}q)$ entrywise,

$$-c \le \max_{j\in[n]} v_j \le 0 \text{ for } c := \log(Cn).$$

Next, we note $\log(\widetilde{Z}q)$ is entrywise bounded in magnitude by $\frac{B}{2}$:

$$\log(\widetilde{Z}q_j) \le \log(CZ) \le \log\left(n \cdot \max_{j\in[n]} \exp([\mathbf{A}^\top x]_j)\right) + \log C \le \frac{B}{2},$$
$$\log(\widetilde{Z}q_j) \ge \log Z + \log\frac{\delta}{n} \ge \min_{j\in[n]}[\mathbf{A}^\top x]_j - \log\frac{n}{\delta} \ge -\frac{B}{2}.$$

Define $\mathbf{M}_1 := \mathbf{diag}\left(\frac{\mathbf{A}^\top x}{2B}\right)$ and $\mathbf{M}_2 := \mathbf{diag}\left(-\frac{1}{2B}\log(\widetilde{Z}q)\right)$. By the calculations above, we have $\|\mathbf{M}_2\|_{\text{op}} \le \frac{1}{2}$, and similarly it is clear that $\|\mathbf{M}_1\|_{\text{op}} \le \frac{1}{2}$ because $\left\|\mathbf{A}^\top x\right\|_\infty \le \beta$. Moreover, by using Corollary B.8 with $\beta \leftarrow B$, we obtain $\mathbf{U}_1$, a block-encoding of $\mathbf{M}_1$ applicable in $O(\log m)$ time. Using a similar construction as Corollary B.8, since q, B, and $\widetilde{Z}$ are all efficiently classically queriable, we obtain $\mathbf{U}_2$, a block-encoding of $\mathbf{M}_2$ applicable in $O(1)$ time. Hence, Proposition B.6 yields $\mathbf{U}$, a block-encoding of

$$\mathbf{M}_1 + \mathbf{M}_2 = \mathbf{diag}\left(\frac{v}{2B}\right),$$

which can be applied in $O(\log mn)$ time. Next, let $\mathcal{P}$ be the degree-$\Delta = O(B\log\frac{Cn}{\delta})$ polynomial from Corollary B.5, parameterized by B, v, c, q, ℓ as defined earlier. Corollary B.5 shows that $\mathcal{P} : [-1, 1] \to [-\frac{1}{2}, \frac{1}{2}]$. Thus, Proposition B.7 then yields $\mathbf{U}'$, a block-encoding of $\mathbf{diag}\left(\mathcal{P}(\frac{v}{2B})\right)$ which can be applied in $O(\Delta \cdot \log mn)$ time. Furthermore, since q and ρ are efficiently classically queriable, we can define a gate $\mathcal{O}_q$ which is applicable in $O(1)$ time and acts as

$$\mathcal{O}_q|0\rangle^{\otimes(b+1)} = |0\rangle \sum_{j\in[n]} \sqrt{\frac{q_j}{\rho}}|j\rangle + |1\rangle|g\rangle.$$

Applying $\mathbf{U}'$ to the output of $\mathcal{O}_q$ with appropriate ancilla qubits then yields

$$|0\rangle^{\otimes O(1)} \sum_{j\in[n]} \sqrt{\frac{q_j\mathcal{P}(u_j)^2}{\rho}}|j\rangle|g_j\rangle + |g'\rangle, \text{ where } u_j := \frac{v_j}{2B} \text{ for all } j \in [n].$$

Post-selecting on the first register being the all-zeroes state and measuring on the register corresponding to j, we see that we obtain a sample $j \in [n]$ with probability proportional to $q_j\mathcal{P}(u_j)^2$. By Corollary B.5, conditioned on the sample succeeding, the resulting distribution is δ-close in ℓ_1 to the distribution proportional to $q \circ \exp(v) \propto \exp(\mathbf{A}^\top x)$, and hence the result is a δ-approximate Gibbs oracle. Finally, we bound the query cost of the oracle. Define $w_j := \mathcal{P}(u_j)^2 q_j$ and $z_j := \exp(v_j)q_j$ as in Corollary B.5. By definition of $v, \widetilde{Z}$,

$$\|z\|_1 = \sum_{j\in[n]} \frac{\exp\left(\left[\mathbf{A}^\top x\right]_j\right)}{\widetilde{Z}} \in \left[C^{-1}, 1\right].$$

Moreover, the last conclusion in Corollary B.5 shows $\|w\|_1 \ge \frac{1}{72}\|z\|_1 \ge (72C)^{-1}$. Hence,

$$\sum_{j\in[n]} \frac{q_j\mathcal{P}(u_j)^2}{\rho} = \frac{\|w\|_1}{\rho} \ge \frac{1}{72C\rho}.$$

In other words, we have an oracle which we can apply in time $O(\Delta \cdot \log mn)$ which correctly returns a sample with probability $\alpha \ge \frac{1}{72C\rho}$. By applying Proposition B.1 to improve the success probability, we obtain the desired conclusion at a $O(\sqrt{C\rho}\log\frac{1}{\delta})$ overhead.

□

C Deferred proofs from Section 3

In this section, we provide proofs for the lemmas and corollaries in Section 3.

Lemma 3.1. *For all $s \in [\lceil \frac{1}{\eta} \rceil]$, $Z_{\tau+s} \in \left[\frac{1}{3}Z_\tau, 3Z_\tau\right]$ and $p_{\tau+s} \in \left[\frac{1}{9}p_\tau, 9p_\tau\right]$ entrywise.*

Proof. Let $\nu_t := \exp(\mathbf{A}^\top x_t)$ for all t, such that $p_t = \frac{\nu_t}{Z_t}$. We have that for any $j \in [n]$,

$$\begin{aligned}\frac{[\nu_{\tau+s}]_j}{[\nu_\tau]_j} &= \exp\left(\left[\mathbf{A}^\top (x_{\tau+s} - x_\tau)\right]_j\right) \\ &\in [\exp\left(-\|\mathbf{A}\|_{\max} \|x_{\tau+s} - x_\tau\|_1\right), \\ &\qquad \exp\left(\|\mathbf{A}\|_{\max} \|x_{\tau+s} - x_\tau\|_1\right)] \\ &\in [\exp(-\eta s), \exp(\eta s)] \in \left[\frac{1}{3}, 3\right].\end{aligned}$$

Similarly, $Z_{\tau+s} \in [\frac{1}{3}Z_\tau, 3Z_\tau]$, and combining yields the conclusion. □

Lemma 3.2. *Let $k \in [n]$, $\alpha \in (0,1)$, and suppose $\delta \le \frac{1}{16k}$. Draw $N = \Theta(k \log \frac{n\eta T}{\alpha})$ samples from $\mathcal{O}_\tau$ for an appropriately large constant, and let $\tilde{q} \in \Delta^n$ be the empirical distribution over these N samples. Define $\mathcal{B} := \{i \in [n] \mid \tilde{q}_i \ge \frac{1}{2k}\}$. Then with probability $\ge 1 - \frac{\alpha}{2\lceil \eta T \rceil}$, for*

$$q_j = \begin{cases} 18\tilde{q}_j & j \in \mathcal{B} \\ \frac{18}{k} & j \notin \mathcal{B} \end{cases},$$

$\|q\|_1 = O(\frac{n}{k})$ and $q \ge p_{\tau+s}$ entrywise, for all $s \le \frac{1}{\eta}$.

Proof. The first conclusion $\|q\|_1 = O(\frac{n}{k})$ is immediate from the definition of q, since $\|q\|_1 \le 18\|\tilde{q}\|_1 + \frac{18n}{k}$. In light of Lemma 3.1 (which holds deterministically), to show the second conclusion, it suffices to show that with the desired success probability, we have both

$$2\tilde{q}_j \ge [p_\tau]_j \text{ for all } j \in \mathcal{B} \tag{15}$$

and

$$\frac{2}{k} \ge [p_\tau]_j \text{ for all } j \notin \mathcal{B}. \tag{16}$$

Denote $\alpha' := \frac{\alpha}{2\lceil \eta T \rceil}$ for notational convenience, and let $\tilde{p}$ denote the distribution of samples from $\mathcal{O}_\tau$, and recall that $\|\tilde{p} - p_\tau\|_1 \le \frac{1}{16k}$. Because we are taking $\Theta(k \log \frac{n}{\alpha'})$ samples from $\tilde{p}$, we have by a standard Chernoff bound that with probability at least $1 - \alpha'$ (union bounding over all coordinates $j \in [n]$), both of the following hold.

1. For all $j \in [n]$ such that $\tilde{p}_j \ge \frac{1}{4k}$, $\tilde{q}_j \ge \frac{2\tilde{p}_j}{3}$.
2. For all $j \in [n]$ such that $\tilde{p}_j \le \frac{1}{4k}$, $\tilde{q}_j \le \frac{1}{2k}$.

Conditioning on these events; we now show (15), (16) in turn.

Proof of (15). To see (15), the second event above implies that if $\tilde{p}_j \le \frac{1}{4k}$, $j \notin \mathcal{B}$. Hence, $\forall j \in \mathcal{B}$, we have $\tilde{q}_j \ge \frac{2\tilde{p}_j}{3} \ge \frac{[p_\tau]_j}{2}$ since $\|\tilde{p} - p_\tau\|_\infty \le \frac{1}{16k} \le \frac{1}{4}\tilde{p}_j, \forall j \in \mathcal{B}$.

Proof of (16). To see (16), suppose for contradiction that $j \notin \mathcal{B}$ and $[p_\tau]_j > \frac{2}{k}$. This implies that $\tilde{p}_j > \frac{1}{k}$, and hence by the first event above, $\tilde{q}_j \ge \frac{1}{2k}$, contradicting $j \notin \mathcal{B}$. □

Corollary 3.3. *Assume Invariants 1, 2 hold for the phase consisting of iterations $\tau + s$, $s \in [\lceil \frac{1}{\eta} \rceil]$. We can solve Problem 1 for the phase with probability $\ge 1 - \frac{\alpha}{2\lceil \eta T \rceil}$, and $\mathcal{T}_{\text{samp}} := O\left(\sqrt{\frac{n}{k}} \cdot T\eta \log^4 \frac{mn}{\delta}\right)$.*

Proof. We will run the algorithm described in the proof of Lemma 3.2, and condition on it succeeding, giving the failure probability. It then suffices to apply Proposition 2.5 with q defined in Lemma 3.2. For this q, we parameterize Proposition 2.5 with $C = O(1)$ (see Invariant 1), $\rho = O(\frac{n}{k})$ (see Lemma 3.2), and $\beta = T\eta$. It is clear the lower bound on entries of q in Proposition 2.5 holds. □

Corollary 3.4. *Following notation of Proposition 2.5, let $R := \frac{\widetilde{Z}}{Z}$. There is a quantum oracle $\mathcal{O}_{\text{test}}$ which can be implemented under T Update calls to x in $O(T \log m)$ time, and has query cost $O(\sqrt{\rho C} \cdot \beta \log^4 \frac{Cmn}{\ell\delta})$. Furthermore, for explicitly known constants C_ℓ and C_u, $\mathcal{O}_{\text{test}}$ returns "success" with probability p for $\frac{C_\ell}{\sqrt{R\rho}} \le p \le \frac{C_u}{\sqrt{R\rho}}$.*

Proof. Our oracle $\mathcal{O}_{\text{test}}$ is the oracle from Proposition 2.5, except we will choose a sufficiently small constant value of δ. It returns "success" when the sample is accepted by the rejection sampler after boosting by amplitude amplification. Before boosting, the success probability from Proposition 2.5 is $\Theta(\frac{1}{R\rho})$ where the constants in the upper and lower bounds are explicit. Further, the constants from Proposition B.1 are explicit, and hence boosting by amplitude amplification improves the success probability to $\Theta(\frac{1}{\sqrt{R\rho}})$ with known constant bounds as required by the corollary statement. $\square$

Lemma 3.5. *Assume Invariants 1, 2 hold for iterations $\tau + s$, $s \in [\lceil \frac{1}{\eta} \rceil]$, and suppose $C \ge \frac{4C_u^2}{C_\ell^2}$ for $C = O(1)$, where C_u and C_ℓ are the constants from Corollary 3.4. Further, suppose we have obtained q satisfying the conclusion of Lemma 3.2 (i.e. that the algorithm in Lemma 3.2 succeeded). We can determine $\widetilde{Z}$ such that $\widetilde{Z} \in [Z_{\tau'}, CZ_{\tau'}]$ with probability $\ge 1 - \frac{\alpha}{2\lceil \eta T \rceil}$, in time $O\left(\sqrt{\frac{n}{k}} \cdot T\eta \log^4 \frac{mn}{\delta} \log \frac{\eta T}{\alpha}\right)$.*

Proof. Define $\widetilde{Z}_0 := 3\widetilde{Z}_{\text{prev}}$, $R_0 := \frac{\widetilde{Z}_0}{Z_{\tau'}}$, and note that $\widetilde{Z}_0 \in [Z_{\tau'}, 9CZ_{\tau'}]$ by Invariant 1 and Lemma 3.1. Next, assuming the success of Lemma 3.2, the success probability p of $\mathcal{O}_{\text{test}}$ from Corollary 3.4 using the estimate $\widetilde{Z}_0$ satisfies (for the unknown $R_0 \in [1, 9C]$, and known C_ℓ, C_u, ρ)

$$\frac{C_\ell}{\sqrt{R_0\rho}} \le p \le \frac{C_u}{\sqrt{R_0\rho}}.$$

For $N := 27 \log \frac{4\lceil \eta T \rceil}{\alpha} \cdot \frac{3\sqrt{C\rho}}{C_\ell}$, we first run $\mathcal{O}_{\text{test}}$ N times and check the number of successes, denoted by S, which fits within the runtime budget by Corollary 3.4. By a Chernoff bound, we have that with probability $\ge 1 - \frac{\alpha}{2\lceil \eta T \rceil}$, we have

$$\begin{aligned} S &\ge \frac{2}{3}pN \ge 54 \log \frac{4\lceil \eta T \rceil}{\alpha} \cdot \sqrt{\frac{C}{R_0}}, \\ S &\le \frac{4}{3}pN \le 108 \log \frac{4\lceil \eta T \rceil}{\alpha} \cdot \frac{C_u}{C_\ell} \cdot \sqrt{\frac{C}{R_0}}. \end{aligned}$$

Hence, we can determine the quantity R_0 up to a multiplicative factor of $\frac{4C_u^2}{C_\ell^2} \le C$, which also implies the same multiplicative approximation factor for $Z_{\tau'}$, as desired. $\square$

D Bounded approximation to $\exp$ on $[-1, 1]$

Here, we give a proof of a lemma (with slightly different constants) used in the prior work (van Apeldoorn & Gilyén, 2019). This section builds entirely off prior results on polynomial approximation in (Gilyén et al., 2019); we include it for completeness because a proof was not given in (van Apeldoorn & Gilyén, 2019). As a reminder, we stated and used the following result earlier when constructing our rejection sampler in Appendix B.

Lemma B.4 (Lemma 7, (van Apeldoorn & Gilyén, 2019))**.** *Let $\beta \ge 1$, $\xi \le \frac{1}{10}$. There is a polynomial $\mathcal{P}_{\beta,\xi}$ of degree $O(\beta \log \frac{1}{\xi})$ such that* $\max_{x \in [-1,1]} |\mathcal{P}_{\beta,\xi}(x)| \le 3$ *and* $\max_{x \in [-1,0]} |\mathcal{P}_{\beta,\xi}(x) - \exp(\beta x)| \le \xi$.

To obtain the lemma, we will utilize the following result from (Gilyén et al., 2019).

Proposition D.1 (Corollary 66, (Gilyén et al., 2019))**.** *Let $x_0 \in [-1, 1]$, $r \in (0, 2]$, $\delta \in (0, r]$. Let $f : [x_0 - r - \delta, x_0 + r + \delta] \to \mathbb{C}$ be such that $f(x_0 + x) = \sum_{\ell \ge 0} a_\ell x^\ell$ for all $x \in [-r - \delta, r + \delta]$. Suppose $B > 0$ is such that $\sum_{\ell \ge 0} (r + \delta)^\ell |a_\ell| \le B$ and let $\epsilon \in (0, \frac{1}{2B}]$. There is a polynomial P (see Appendix E for its numerically stable implementation) of degree $O\left(\frac{1}{\delta} \log \frac{B}{\epsilon}\right)$ such that*

$$\max_{x \in [x_0 - r, x_0 + r]} |f(x) - P(x)| \le \epsilon \text{ and } \max_{x \in [-1,1]} |P(x)| \le \epsilon + B.$$

Proof of Lemma B.4. We apply Proposition D.1 with $f(x) := \exp(\beta x)$ which has a convergent Taylor series everywhere, and the parameter settings $x_0 = -1$, $r = 1$, $\delta = \frac{1}{\beta}$, $B = e$. We have that $f(x_0 + x) = \sum_{\ell \geq 0} \exp(-\beta) \frac{\beta^\ell \cdot x^\ell}{\ell!} = \sum_{\ell \geq 0} a_\ell x^\ell$ with $a_\ell = \exp(-\beta) \frac{\beta^\ell}{\ell!}$ for any integer $\ell \geq 0$. We also check that our choice of B is valid, via

$$\sum_{\ell \geq 0} (r+\delta)^\ell |a_\ell| = \exp(-\beta) \sum_{\ell \geq 0} \left(1 + \frac{1}{\beta}\right)^\ell \frac{\beta^\ell}{\ell!} = \exp(-\beta) \sum_{\ell \geq 0} \frac{(\beta+1)^\ell}{\ell!} = \exp(\beta + 1 - \beta) = e.$$

Hence by Proposition D.1, we have for any $\xi \leq \frac{1}{2e}$, there is a polynomial P of degree $O(\beta \log \frac{1}{\xi})$ such that $\max_{x \in [-2,0]} |\exp(\beta x) - P(x)| \leq \epsilon$ and $\max_{x \in [-1,1]} |\tilde{P}(x)| \leq e + \frac{1}{6} + \xi \leq 3$. □

E Numerically stable implementation of polynomial approximation

In this section, we explain the derivation of the additive $\widetilde{O}(\text{poly}(\epsilon^{-1}))$ term in Theorem 2.3. When applying Proposition 2.5, we assumed access to an implementation of the polynomial transform of our block-encoded matrix through the framework of (Gilyén et al., 2019), whose complexity is bounded in that work. However, using the implementation in (Gilyén et al., 2019) also requires that we have computed certain phase factors associated with the polynomial. These phase factors are not available in closed form and hence must be numerically approximated, which we now discuss. Throughout, we let $\Delta = O(\frac{1}{\epsilon} \log^2(\frac{mn}{\epsilon}))$ be the degree of the polynomial used in the proof of Proposition 2.5 in Appendix B (specifically, constructed in the proof of Proposition 2.5, where we have $C = O(1)$ and $\delta = O(\epsilon)$ in our applications). Our polynomial is constructed via a decomposition in the Fourier basis (see Lemmas 57 and 65, (Gilyén et al., 2019)). Furthermore, (Haah, 2019) shows that given such a decomposition in the Fourier basis, we can obtain a numerically-stable implementation of the phase factors required resulting in a quantum circuit implementing the desired polynomial up to additive error ξ, in time

$$O\left(\Delta^3 \log\left(\frac{\Delta}{\xi}\right)\right).$$

In our setting (in the proof of Proposition 2.5), it is straightforward to check that $\frac{1}{\xi} = \text{poly}(m, n, \epsilon^{-1})$. This construction hence results in the additive term in Theorem 2.3.