Quantum Algorithms for Matrix Scaling and Matrix Balancing

Joran van Apeldoorn
Institute for Information Law and QuSoft, University of Amsterdam, The Netherlands

Sander Gribling
IRIF, Université de Paris, CNRS, Paris, France

Yinan Li
Graduate School of Mathematics, Nagoya University, Japan

Harold Nieuwboer
Korteweg–de Vries Institute for Mathematics and QuSoft,
University of Amsterdam, The Netherlands

Michael Walter
KdVI, ITFA, ILLC, and QuSoft, University of Amsterdam, The Netherlands

Ronald de Wolf
QuSoft, CWI, Amsterdam, The Netherlands
University of Amsterdam, The Netherlands

Abstract

Matrix scaling and matrix balancing are two basic linear-algebraic problems with a wide variety of applications, such as approximating the permanent, and pre-conditioning linear systems to make them more numerically stable. We study the power and limitations of quantum algorithms for these problems. We provide quantum implementations of two classical (in both senses of the word) methods: Sinkhorn's algorithm for matrix scaling and Osborne's algorithm for matrix balancing. Using amplitude estimation as our main tool, our quantum implementations both run in time $\widetilde{O}(\sqrt{mn}/\varepsilon^4)$ for scaling or balancing an $n \times n$ matrix (given by an oracle) with m non-zero entries to within ℓ_1-error ε. Their classical analogs use time $\widetilde{O}(m/\varepsilon^2)$, and every classical algorithm for scaling or balancing with small constant ε requires $\Omega(m)$ queries to the entries of the input matrix. We thus achieve a polynomial speed-up in terms of n, at the expense of a worse polynomial dependence on the obtained ℓ_1-error ε. Even for constant ε these problems are already non-trivial (and relevant in applications). Along the way, we extend the classical analysis of Sinkhorn's and Osborne's algorithm to allow for errors in the computation of marginals. We also adapt an improved analysis of Sinkhorn's algorithm for entrywise-positive matrices to the ℓ_1-setting, obtaining an $\widetilde{O}(n^{1.5}/\varepsilon^3)$-time quantum algorithm for ε-ℓ_1-scaling. We also prove a lower bound, showing our quantum algorithm for matrix scaling is essentially optimal for constant ε: every quantum algorithm for matrix scaling that achieves a constant ℓ_1-error w.r.t. uniform marginals needs $\Omega(\sqrt{mn})$ queries.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Quantum computation theory

Keywords and phrases Matrix scaling, matrix balancing, quantum algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.110

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2011.12823v1` [9]

Funding *Joran van Apeldoorn*: Dutch Research Council (NWO/OCW), as part of QSC (024.003.037).
Sander Gribling: Partially supported by SIRTEQ-grant QuIPP.
Yinan Li: MEXT Quantum Leap Flagship Program grant no. JPMXS0120319794.
Harold Nieuwboer: NWO grant no. OCENW.KLEIN.267.
Michael Walter: NWO Veni grant no. 680-47-459 and grant no. OCENW.KLEIN.267.
Ronald de Wolf: NWO/OCW through QSC (024.003.037) and QuantAlgo (QuantERA 680-91-034).

Acknowledgements We thank the ICALP referees for some very helpful feedback.

1 Introduction

1.1 Matrix scaling and matrix balancing

Matrix scaling is a basic linear-algebraic problem with many applications. A *scaling* of an $n \times n$ matrix $\mathbf{A}$ with non-negative entries is a matrix $\mathbf{B} = \mathbf{XAY}$ where $\mathbf{X}$ and $\mathbf{Y}$ are positive diagonal matrices (everything straightforwardly extends to non-square $\mathbf{A}$). In other words, we multiply the i-th row with X_{ii} and the j-th column with Y_{jj}. We say $\mathbf{A}$ is *exactly scalable* to marginals $\mathbf{r} \in \mathbb{R}^n_+$ and $\mathbf{c} \in \mathbb{R}^n_+$ if there exist $\mathbf{X}$ and $\mathbf{Y}$ such that the vector $\mathbf{r}(\mathbf{B}) = (\sum_{j=1}^n B_{ij})_{i\in[n]}$ of row sums of the scaled matrix $\mathbf{B}$ equals $\mathbf{r}$, and its vector $\mathbf{c}(\mathbf{B})$ of column sums equals $\mathbf{c}$. One typical example would be if $\mathbf{r}$ and $\mathbf{c}$ are the all-1 vectors, which means we want $\mathbf{B}$ to be doubly stochastic: the rows and columns of $\mathbf{B}$ would be probability distributions. In many cases it suffices to find *approximate* scalings. Different applications use different notions of approximation. We could for instance require $\mathbf{r}(\mathbf{B})$ to be ε-close to $\mathbf{r}$ in ℓ_1- or ℓ_2-norm, or in relative entropy (Kullback-Leibler divergence), for some parameter ε of our choice, and similarly require $\mathbf{c}(\mathbf{B})$ to be ε-close to $\mathbf{c}$.

A related problem is *matrix balancing*. Here we do not prescribe desired marginals, but the goal is to find a diagonal $\mathbf{X}$ such that the row and column marginals of $\mathbf{B} = \mathbf{XAX^{-1}}$ are close *to each other*. Again, different notions of closeness $\mathbf{r}(\mathbf{B}) \approx \mathbf{c}(\mathbf{B})$ are possible.

An important application, used in theory as well as in practical linear-algebra software (e.g. LAPACK [6] and MATLAB [40]), is in improving the numerical stability of linear-system solving. Suppose we are given matrix $\mathbf{A}$ and vector $\mathbf{b}$, and we want to find a solution to the linear system $\mathbf{Av} = \mathbf{b}$. Note that $\mathbf{v}$ is a solution iff $\mathbf{Bv}' = \mathbf{b}'$ for $\mathbf{v}' = \mathbf{Xv}$ and $\mathbf{b}' = \mathbf{Xb}$. An appropriately balanced matrix $\mathbf{B}$ will typically be more numerically stable than the original $\mathbf{A}$, so solving the linear system $\mathbf{Bv}' = \mathbf{b}'$ and then computing $\mathbf{v} = \mathbf{X}^{-1}\mathbf{v}'$, is often a better way to solve the linear system $\mathbf{Av} = \mathbf{b}$ than directly computing $\mathbf{A}^{-1}\mathbf{b}$.

Matrix scaling and balancing have surprisingly many and wide-ranging applications. Matrix scaling was introduced by Kruithof for Dutch telephone traffic computation [37], and has also been used in other areas of economics [50]. In theoretical computer science it has been used for instance to approximate the permanent of a given matrix [38], as a tool to get lower bounds on unbounded-error communication complexity [25], and for approximating optimal transport distances [2]. In mathematics, it has been used as a common tool in practical linear algebra computations [39, 12, 46, 42], but also in statistics [49], optimization [47], and for strengthening the Sylvester-Gallai theorem [11]. Matrix balancing has a similarly wide variety of applications, including pre-conditioning to make practical matrix computations more stable (as mentioned above), and approximating the min-mean-cycle in a weighted graph [3]. Many more applications of matrix scaling and balancing are mentioned in [38, 31, 28]. Related scaling problems have applications to algorithmic non-commutative algebra [27, 17], functional analysis [26], and quantum information [29, 18, 16].

1.2 Known (classical) algorithms

Given the importance of good matrix scalings and balancings, how efficiently can we actually find them? For concreteness, let us first focus on scaling. Note that left-multiplying $\mathbf{A}$ with a diagonal matrix $\mathbf{X}$ corresponds to multiplying the i-th row of $\mathbf{A}$ with X_{ii}. Hence it is very easy to get the desired row sums: just compute all row sums $r_i(\mathbf{A})$ of $\mathbf{A}$ and define $\mathbf{X}$ by $X_{ii} = r_i/r_i(\mathbf{A})$, then $\mathbf{XA}$ has exactly the right row sums. Subsequently, it is easy to get the desired column sums: just right-multiply the current matrix $\mathbf{XA}$ with diagonal matrix $\mathbf{Y}$ where $Y_{jj} = c_j/c_j(\mathbf{XA})$, then $\mathbf{XAY}$ will have the right column sums.

The problem with this approach, of course, is that the second step is likely to undo the good work of the first step, changing the row sums away from the desired values; it is not at all obvious how to *simultaneously* get the row sums and column sums right. Nevertheless, the approach of alternating row-normalizations with column-normalizations turns out to work. This alternating algorithm is known as *Sinkhorn's algorithm* [49], and has actually been (re)discovered independently in several different contexts.

For matrix balancing there is a similar method called *Osborne's algorithm* [43, 45]. In each iteration this chooses a row index i and defines X_{ii} such that the i-th row sum and the i-th column sum become equal. Again, because each iteration can undo the good work of earlier iterations, convergence to a balancing of $\mathbf{A}$ is not at all obvious. Remarkably, even though Osborne's algorithm was proposed more than six decades ago and is widely used in linear algebra software, an explicit convergence-rate bound was only proved recently [48, 44]!

At the same time there have been other, more sophisticated algorithmic approaches for scaling and balancing. Just to mention one: we can parametrize $\mathbf{X} = \operatorname{diag}(e^{\mathbf{x}})$ and $\mathbf{Y} = \operatorname{diag}(e^{\mathbf{y}})$ by vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and consider the following convex potential function:

$$f(\mathbf{x}, \mathbf{y}) = \sum_{i,j=1}^{n} A_{ij} e^{x_i + y_j} - \sum_{i=1}^{n} r_i x_i - \sum_{j=1}^{n} c_j y_j.$$

Note that the partial derivative of this f w.r.t. the variable x_i is $\sum_{j=1}^{n} A_{ij} e^{x_i+y_j} - r_i = r_i(\mathbf{XAY}) - r_i$, and the partial derivative w.r.t. y_j is $c_j(\mathbf{XAY}) - c_j$. A minimizer $\mathbf{x}, \mathbf{y}$ of f will have the property that all these $2n$ partial derivatives are equal to 0, which means $\mathbf{XAY}$ *is exactly scaled!* Accordingly, (approximate) scalings can be obtained by finding (approximate) minimizers using methods from convex optimization. In fact, Sinkhorn's original algorithm can be interpreted as block coordinate descent on this f, and Osborne's algorithm can similarly be derived by slightly modifying f. More advanced methods from convex optimization have also been applied, such as ellipsoid methods [36, 33, 41], box-constrained Newton methods [1, 21] and interior-point methods [21, 19].

Historically, research on matrix scaling and matrix balancing (and generalizations such as operator scaling) has focused on finding ε-ℓ_2-scalings. More recently also algorithms for finding ε-ℓ_1-scalings have been extensively studied, due to their close connection with permanents and finding perfect matchings in bipartite graphs [38, 20], and because the ℓ_1-distance is an important error measure for statistical problems such as computing the optimal transport distance between distributions [22, 2], even already for constant ε. By the Cauchy-Schwarz inequality, an $(\varepsilon/\sqrt{n})$-ℓ_2-scaling for $\mathbf{A}$ is also an ε-ℓ_1-scaling, but often more direct methods work better for finding an ε-ℓ_1-scaling.

Below in Table 1 we tabulate the best known algorithms for finding ε-scalings in ℓ_1-norm for entrywise-positive matrices and general non-negative matrices. We make the standard assumptions that every entry of the target marginals $\mathbf{r}, \mathbf{c}$ is non-zero, and that $\mathbf{A}$ is *asymptotically scalable*: for every $\varepsilon > 0$, there exist $\mathbf{X}$ and $\mathbf{Y}$ such that

$$\|\mathbf{r}(\mathbf{B}) - \mathbf{r}\|_1 \leq \varepsilon \text{ and } \|\mathbf{c}(\mathbf{B}) - \mathbf{c}\|_1 \leq \varepsilon,$$

where $\mathbf{B} = \mathbf{XAY}$ (this implies that the matrix has at least one non-zero entry in every row and column). A sufficient condition for this is that the matrix is entrywise-positive. To state the complexity results, let m be the number of non-zero entries in $\mathbf{A}$ (note that $m \geq n$), assume $\sum_{i,j=1}^{n} A_{ij} = 1$, that its non-zero entries lie in $[\mu, 1]$, and $\|\mathbf{r}\|_1 = \|\mathbf{c}\|_1 = 1$ (so uniform marginal is $\mathbf{1}/n$). We will assume $\varepsilon \in (0, 1)$. The input numbers to the algorithm are all assumed to be rational, with bit size bounded by $\operatorname{polylog}(n)$, unless specified otherwise.[1]

[1] The complexity of our algorithms depends polylogarithmically on the magnitude of the entries; the assumption on the bit size of the entries is made to simplify the presentation.

Note that the table contains both first-order and second-order methods; the former just use the gradient of the potential (or a related potential), whereas the latter also use its Hessian. The second-order methods typically have a polylogarithmic dependence on the inverse of the desired precision ε, whereas the first-order methods have inverse polynomial dependence on ε. For entrywise-positive matrices, second-order methods theoretically outperform the *classical* first-order methods in any parameter regime. However, they depend on non-trivial results for graph sparsification and Laplacian system solving which are relatively complicated to implement, in contrast to the eminently practical (first-order) Sinkhorn and Osborne.

For matrix balancing, Osborne's algorithm has very recently been shown to produce an ε-ℓ_1-balancing in time $\widetilde{O}(m/\varepsilon^2)$ when in each iteration the update is chosen randomly [4]. Algorithms based on box-constrained Newton methods and interior-point methods can find ε-balancings in time $\widetilde{O}(m \log \kappa)$ and $\widetilde{O}(m^{1.5})$, respectively, where κ denotes the ratio between the largest and the smallest entries of the optimal balancing.

Table 1 State-of-the-art time complexity of first- and second-order methods for finding an ε-ℓ_1-scaling, both to uniform marginals and to arbitrary marginals. The boldface lines are from this paper, and the only quantum algorithms for scaling we are aware of. h is the smallest integer such that $h\mathbf{r}$, $h\mathbf{c}$ are integer vectors; m upper bounds the number of non-zero entries of $\mathbf{A}$; κ is the ratio between largest and smallest entries of the optimal scalings $\mathbf{X}$ and $\mathbf{Y}$, which can be exponential in n. Many references use a different error model (e.g., ℓ_2 or Kullback-Leibler), which we convert to guarantees in ℓ_1-norm for comparison. $\widetilde{O}$-notation hides polylogarithmic factors in n, $1/\varepsilon$, $1/\mu$.

	$(\mathbf{1}/n, \mathbf{1}/n)$	$(\mathbf{r}, \mathbf{c})$	References and remarks
General non-negative	$\widetilde{O}(m/\varepsilon^2)$	$\widetilde{O}(m/\varepsilon^2)$	Sinkhorn, via KL [20][2]
	$\widetilde{O}(mn^{2/3}/\varepsilon^{2/3})$	$\widetilde{O}(mn/(h^{1/3}\varepsilon^{2/3}))$	first-order, via ℓ_2 [1]
	$\widetilde{O}(m \log \kappa)$	$\widetilde{O}(m \log \kappa)$	box-constrained method, via ℓ_2 [21]
	$\widetilde{O}(m^{1.5})$	$\widetilde{O}(m^{1.5})$	interior-point method, via ℓ_2 [21]
	$\widetilde{O}(\sqrt{mn}/\varepsilon^4)$	$\widetilde{O}(\sqrt{mn}/\varepsilon^4)$	**Sinkhorn, quantum, Corollary 5**
Entrywise positive	$\widetilde{O}(n^2/\varepsilon)$	$\widetilde{O}(n^3/\varepsilon)$	Sinkhorn, via ℓ_2 [35, 34], $h\varepsilon \leq \sqrt{2n}$
	$\widetilde{O}(n^2/\varepsilon^2)$	$\widetilde{O}(n^2/\varepsilon^2)$	Sinkhorn, via KL [2, 20]
	$\widetilde{O}(n^2)$	$\widetilde{O}(n^2)$	box-constrained, via ℓ_2 [1, 21]
	$\widetilde{O}(n^{1.5}/\varepsilon^3)$	$\widetilde{O}(n^{1.5}/\varepsilon^3)$	**Sinkhorn, quantum, Corollary 7**

1.3 First contribution: quantum algorithms for ℓ_1-scaling and balancing

Because a classical scaling algorithm has to look at each non-zero matrix entry (at least with large probability), it is clear that $\Omega(m)$ is a classical lower bound. This is $\Omega(n^2)$ in the case of a dense or even entrywise-positive matrix $\mathbf{A}$. As can be seen from Table 1, the best classical algorithms also achieve this m lower bound up to logarithmic factors, with various dependencies on ε. The same is true for balancing: $\Omega(m)$ queries are necessary, and this is achievable in different ways, with different dependencies on ε and other parameters.

Our first contribution is to give (in Section 3) quantum algorithms for scaling and balancing that beat the best-possible classical algorithms, at least for relatively large $\varepsilon \in (0, 1)$:

[2] Their proofs work only for input matrices that are exactly scalable. However, with our potential gap bound we can generalize their analysis to work for arbitrary asymptotically-scalable matrices.

Scaling: We give a quantum algorithm that (with probability $\geq 2/3$) finds an ε-ℓ_1-scaling for an asymptotically-scalable $n \times n$ matrix $\mathbf{A}$ with m non-zero entries (given by an oracle) to desired positive marginals $\mathbf{r}$ and $\mathbf{c}$ in time $\widetilde{O}(\sqrt{mn}/\varepsilon^4)$. When $\mathbf{A}$ is entrywise positive (so $m = n^2$), the upper bound can be improved to $\widetilde{O}(n^{1.5}/\varepsilon^3)$.

Balancing: We give a quantum algorithm that (with probability $\geq 2/3$) finds an ε-ℓ_1-balancing for an asymptotically-balanceable $n \times n$ matrix $\mathbf{A}$ with m non-zero entries (given by an oracle) in time $\widetilde{O}(\sqrt{mn}/\varepsilon^4)$.

Our scaling algorithms in fact achieve closeness measured in terms of the relative entropy, and then use Pinsker's inequality ($\|\mathbf{p} - \mathbf{q}\|_1^2 = O(D(\mathbf{p}||\mathbf{q}))$) to convert this to an upper bound on the ℓ_1-error. Our algorithms achieve a *sublinear* dependence on the input size m.

Note that compared to the classical algorithms we have polynomially better dependence on n and m, at the expense of a worse dependence on ε. There have recently been a number of new quantum algorithms with a similar tradeoff: they are better than classical in terms of the main size parameter but worse in terms of the precision parameter. Examples are the quantum algorithms for solving linear and semidefinite programs [14, 8, 13, 7] and for boosting of weak learning algorithms [10, 30, 32].

Conceptually our algorithms are quite simple: we implement the Sinkhorn and Osborne algorithms but replace the exact computation of each row and column sum by *quantum amplitude estimation*; this allows us to approximate the sum of n numbers up to some small multiplicative error δ (with high probability) at the expense of roughly $\sqrt{n}/\delta$ queries to those numbers, and a similar number of other operations.

Our analysis is based on a potential argument (for Sinkhorn we use the above-mentioned potential f). The error δ causes us to make less progress in each iteration compared to an "exact" version of Sinkhorn or Osborne. If δ is too large we may even make backwards progress, while if δ is very small there is no quantum speed-up! We show there is a choice of δ for which the negative contribution due to the approximation errors is of the same order as the progress in the "exact" version, and that choice results in a speed-up. We should caution, however, that it is quite complicated to actually implement this idea precisely and to keep track of and control the various approximation errors and error probabilities induced by the quantum estimation algorithms, as well as by the fact that we cannot represent the numbers involved with infinite precision (this issue of precision is sometimes swept under the rug in classical research on scaling algorithms). Finally, we note that due to the error δ our potential need not decrease monotonically. The standard analysis of Sinkhorn still applies if we can *test* whether the current scaling is an ε-scaling after each full Sinkhorn iteration. We show how to do so efficiently in the quantum setting. However, in Osborne's algorithm one updates only a single (random) row/column per iteration, and the quantum cost of our testing procedure is higher than the cost of updating (classically, this problem is overcome by simply keeping track of the row and column marginals). To circumvent the need for testing every iteration, we give a novel analysis of Osborne's algorithm (and of a randomized version of Sinkhorn) showing uniformly random iterates provide an ε-balancing with high probability.

1.4 Second contribution: quantum lower bound for scaling

A natural question would be whether our upper bounds for the time complexity of matrix scaling and balancing can be improved further. Since the output has length roughly n, there is an obvious lower bound of n even for quantum algorithms. An $\widetilde{O}(n)$ quantum algorithm would, however, still be an improvement over our algorithms, and it would be a quadratic speed-up over the best classical algorithm. In Section 4 we dash this hope for matrix scaling by showing that our algorithm is essentially optimal for constant ε, even for the special case of $\mathbf{A}$ that is exactly scalable to uniform marginals:

> There exists a constant $\varepsilon > 0$ such that every quantum algorithm that (with probability $\geq 2/3$) finds an ε-ℓ_1-scaling for a given $n \times n$ matrix $\mathbf{A}$ that is exactly scalable to uniform and has m potentially non-zero entries, has to make $\Omega(\sqrt{mn})$ queries to $\mathbf{A}$.

Our proof constructs instances $\mathbf{A}$ that hide permutations, shows how approximate scalings of $\mathbf{A}$ give us information about the hidden permutation, and then uses the adversary method [5] to lower bound the number of quantum queries to the matrix needed to find that information. In particular, we show that for a permutation $\sigma \in S_n$, it takes $\Omega(n\sqrt{n})$ queries to the entries of the associated permutation matrix to learn $\sigma(i) \bmod 2$ for each $i \in [n]$.

2 Preliminaries

2.1 Matrix scaling and balancing

Throughout we use $\mathbf{r}, \mathbf{c} \in \mathbb{R}^n$ as the desired row and column marginals. Unambiguously, we also use $\mathbf{r}\colon \mathbb{R}^{n\times n} \to \mathbb{R}^n$ (resp. $\mathbf{c}\colon \mathbb{R}^{n\times n} \to \mathbb{R}^n$) as the function that sends an $n \times n$-matrix to its row (resp. column) marginal: $\mathbf{r}(\mathbf{A})$ (resp. $\mathbf{c}(\mathbf{A})$) is the vector whose i-th entry equals $r_i(\mathbf{A}) = \sum_{j=1}^n A_{ij}$ (resp. $c_i(\mathbf{A}) = \sum_{j=1}^n A_{ji}$). We use $\mathbf{A}(\mathbf{x},\mathbf{y}) = (A_{ij}e^{x_i+y_j})_{i,j\in[n]}$ to denote the rescaled matrix $\mathbf{A}$ with scalings given by $e^{\mathbf{x}}$ and $e^{\mathbf{y}}$. We say a non-negative matrix $\mathbf{A} \in \mathbb{R}_+^{n\times n}$ is *exactly scalable* to $(\mathbf{r},\mathbf{c}) \in \mathbb{R}_+^n \times \mathbb{R}_+^n$, if there exist $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$ such that $\mathbf{r}(\mathbf{A}(\mathbf{x},\mathbf{y})) = \mathbf{r}$ and $\mathbf{c}(\mathbf{A}(\mathbf{x},\mathbf{y})) = \mathbf{c}$. For an $\varepsilon > 0$, we say $\mathbf{A} \in \mathbb{R}_+^{n\times n}$ is *ε-ℓ_1-scalable* to $(\mathbf{r},\mathbf{c}) \in \mathbb{R}_+^n \times \mathbb{R}_+^n$, if there exist $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$ such that $\|\mathbf{r}(\mathbf{A}(\mathbf{x},\mathbf{y})) - \mathbf{r}\|_1 \leq \varepsilon$ and $\|\mathbf{c}(\mathbf{A}(\mathbf{x},\mathbf{y})) - \mathbf{c}\|_1 \leq \varepsilon$. We say $\mathbf{A} \in \mathbb{R}_+^{n\times n}$ is *asymptotically scalable* to $(\mathbf{r},\mathbf{c}) \in \mathbb{R}_+^n \times \mathbb{R}_+^n$ if it is ε-ℓ_1-scalable to $(\mathbf{r},\mathbf{c})$ for every $\varepsilon > 0$. In the matrix-balancing setting we require $\mathbf{y} = -\mathbf{x}$, and the marginals are compared to each other. We abbreviate $\mathbf{A}(\mathbf{x}) = \mathbf{A}(\mathbf{x},-\mathbf{x})$. We say a non-negative matrix $\mathbf{A} \in \mathbb{R}_+^{n\times n}$ is *exactly balanceable*, if there exists a vector $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{r}(\mathbf{A}(\mathbf{x})) = \mathbf{c}(\mathbf{A}(\mathbf{x}))$. For an $\varepsilon > 0$, we say $\mathbf{A} \in \mathbb{R}_+^{n\times n}$ is *ε-ℓ_1-balanceable*, if there exists an $\mathbf{x} \in \mathbb{R}^n$ such that $\frac{\|\mathbf{r}(\mathbf{A}(\mathbf{x}))-\mathbf{c}(\mathbf{A}(\mathbf{x}))\|_1}{\|\mathbf{A}(\mathbf{x})\|_1} \leq \varepsilon$. We say $\mathbf{A} \in \mathbb{R}_+^{n\times n}$ is *asymptotically balanceable* if it is ε-ℓ_1-balanceable for every $\varepsilon > 0$. The associated optimization problems are as follows.

▶ **Problem 1** (ε-ℓ_1-scaling problem)**.** *Given $\mathbf{A} \in \mathbb{R}_+^{n\times n}$ and desired marginals $\mathbf{r},\mathbf{c} \in \mathbb{R}_+^n$ with $\|\mathbf{r}\|_1 = \|\mathbf{c}\|_1 = 1$, find $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$ s.t. $\|\mathbf{r}(\mathbf{A}(\mathbf{x},\mathbf{y})) - \mathbf{r}\|_1 \leq \varepsilon$ and $\|\mathbf{c}(\mathbf{A}(\mathbf{x},\mathbf{y})) - \mathbf{c}\|_1 \leq \varepsilon$.*

▶ **Problem 2** (ε-ℓ_1-balancing problem)**.** *Given $\mathbf{A} \in \mathbb{R}_+^{n\times n}$, find $\mathbf{x} \in \mathbb{R}^n$ s.t. $\|\mathbf{r}(\mathbf{A}(\mathbf{x})) - \mathbf{c}(\mathbf{A}(\mathbf{x}))\|_1/\|\mathbf{A}(\mathbf{x})\|_1 \leq \varepsilon$.*

For matrix scaling, our algorithm is most naturally analyzed with the error measured by the relative entropy, which can be converted to give an upper bound on ℓ_1-distance using (a generalized version of) Pinsker's inequality. We therefore also consider the following problem:

▶ **Problem 3** (ε-relative-entropy-scaling problem)**.** *Given $\mathbf{A} \in \mathbb{R}_+^{n\times n}$ and desired marginals $\mathbf{r},\mathbf{c} \in \mathbb{R}_+^n$ with $\|\mathbf{r}\|_1 = \|\mathbf{c}\|_1 = 1$, find $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$ such that $D(\mathbf{r}\|\mathbf{r}(\mathbf{A}(\mathbf{x},\mathbf{y}))) \leq \varepsilon$ and $D(\mathbf{c}\|\mathbf{c}(\mathbf{A}(\mathbf{x},\mathbf{y}))) \leq \varepsilon$.*

2.2 Computational model

We assume *sparse black-box access* to the elements of $\mathbf{A}$ via lists of the potentially non-zero entries for each row and each column. A quantum algorithm can make such queries also in superposition. We also assume (classical) black-box access to the target marginals $\mathbf{r}, \mathbf{c}$. Our computational model is of a classical computer (say, a Random Access Machine for concreteness) that can invoke a quantum computer as a subroutine. The classical computer can write to a classical-write quantum-read memory ("QCRAM")[3], and send a description of a quantum circuit that consists of one- and two-qubit gates from some fixed discrete universal gate set (say, the H and T gates, Controlled-NOT, and 2-qubit controlled rotations over angles $2\pi/2^s$ for positive integers s; these controlled rotations are used in the circuit for the quantum Fourier transform (QFT), which we invoke later), queries to the input oracles, and queries to the QCRAM to the quantum computer. The quantum computer runs the circuit, measures the full final state in the computational basis, and returns the measurement outcome to the classical computer. See [9, Sec. 2] for details.

3 A Sinkhorn algorithm with approximate updates

We state and analyze Algorithm 1, a variant of the well-known Sinkhorn algorithm. Here we give an overview of its analysis, we refer to [9, Sec. 3] for the proofs. The algorithm's objective is to find scaling vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ such that the matrix $\mathbf{A}(\mathbf{x}, \mathbf{y}) = (A_{ij}e^{x_i+y_j})_{i,j\in[n]}$ has row and column marginals $\mathbf{r}$ and $\mathbf{c}$, respectively. Sinkhorn-type algorithms do so in the following iterative way. Starting from the rational matrix $\mathbf{A} \in [0,1]^{n\times n}$, find a vector $\mathbf{x}$ such that the row marginals of $(A_{ij}e^{x_i})_{i,j\in[n]}$ are $\mathbf{r}$, and then find a $\mathbf{y}$ such that the column marginals of $\mathbf{A}(\mathbf{x}, \mathbf{y})$ are $\mathbf{c}$. The second step may have changed the row marginals, so we repeat the procedure. We can view this as updating the coordinates of $\mathbf{x}$ and $\mathbf{y}$ one at a time, starting from the all-0 vectors. To update the row scaling vectors, we wish to find $\hat{\mathbf{x}} = \mathbf{x} + \mathbf{\Delta}$ such that $\mathbf{r}(\mathbf{A}(\hat{\mathbf{x}}, \mathbf{y})) = \mathbf{r}$. Expanding this equation yields $e^{\Delta_\ell} \cdot r_\ell(\mathbf{A}(\mathbf{x}, \mathbf{y})) = r_\ell$, for $\ell \in [n]$. Every row and column contains at least one non-zero entry, so this has a unique solution:

$$\hat{x}_\ell = x_\ell + \Delta_\ell = x_\ell + \ln\left(\frac{r_\ell}{r_\ell(\mathbf{A}(\mathbf{x}, \mathbf{y}))}\right) = \ln\left(\frac{r_\ell}{\sum_{j=1}^n A_{\ell j}e^{y_j}}\right). \tag{3.1}$$

Similar formulas can be derived for the column-updates. We use the term "one Sinkhorn iteration" to refer to the process of updating all n row scaling vectors, or updating all n column scaling vectors. We state the Sinkhorn algorithm in terms of two subroutines, `ApproxScalingFactor` and `TestScaling`. For both subroutines we provide both classical and quantum implementations in [9, Sec. 4]. A key ingredient of both subroutines is a procedure that computes the logarithm of a sum of exponentials, see Section 3.2 for a high-level explanation of the quantum subroutine. For the analysis of Algorithm 1, we only use the guarantees of the subroutines as stated, and do not refer to their actual implementation.

We study a version of Sinkhorn's algorithm where, instead of computing row and column marginals in each iteration exactly, we use a *multiplicative* approximation of the marginals to compute δ-additive approximations of Equation (3.1) and similar for column-updates. In the classical literature, δ can be chosen to be very small, since the cost per iteration scales as $\operatorname{polylog}(1/\delta)$, and hence that error is essentially a minor technical detail. In the quantum setting, we obtain better dependence in terms of n at the cost of allowing a $\operatorname{poly}(1/\delta)$-dependence, so the required precision δ merits detailed attention in the analysis.

[3] Note that we do not require a full QRAM that can hold qubits as well. We believe QCRAM is a natural assumption since it simply amounts to classical RAM that can be read in superposition.

Algorithm 1 Full Sinkhorn with finite precision and failure probability.

```
Input: Oracle access to $\mathbf{A} \in [0,1]^{n\times n}$ with $\|\mathbf{A}\|_1 \leq 1$ and non-zero entries at least
       $\mu > 0$, target marginals $\mathbf{r}, \mathbf{c} \in (0,1]^n$ with $\|\mathbf{r}\|_1 = \|\mathbf{c}\|_1 = 1$, iteration
       count $T \in \mathbb{N}$, bit counts $b_1, b_2 \in \mathbb{N}$, estimation precision $0 < \delta < 1$, test
       precision $0 < \delta' < 1$ and subroutine failure probability $\eta \in [0,1]$
Output: Vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ with entries encoded in $(b_1, b_2)$ fixed-point format
Guarantee: For $\varepsilon \in (0,1]$, with parameters chosen as in Proposition 13, $(\mathbf{x}, \mathbf{y})$ form
           an $\varepsilon$-relative-entropy-scaling of $\mathbf{A}$ to $(\mathbf{r}, \mathbf{c})$ with probability $\geq 2/3$
$\mathbf{x}^{(0)}, \mathbf{y}^{(0)} \leftarrow \mathbf{0}$;            // entries in $(b_1, b_2)$ fixed-point format
for $t \leftarrow 1, 2, \ldots, T$ do
    if $t$ is odd then
        for $\ell \leftarrow 1, 2, \ldots, n$ do
            $x_\ell^{(t)} \leftarrow$ ApproxScalingFactor($\mathbf{A}_{\ell\bullet}, r_\ell, \mathbf{y}^{(t-1)}, \delta, b_1, b_2, \eta, \mu$);
        end for
        $\mathbf{y}^{(t)} \leftarrow \mathbf{y}^{(t-1)}$;
    else if $t$ is even then
        for $\ell \leftarrow 1, 2, \ldots, n$ do
            $y_\ell^{(t)} \leftarrow$ ApproxScalingFactor($\mathbf{A}_{\bullet\ell}, c_\ell, \mathbf{x}^{(t-1)}, \delta, b_1, b_2, \eta, \mu$);
        end for
        $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)}$;
    end if
    if TestScaling($\mathbf{A}, \mathbf{r}, \mathbf{c}, \mathbf{x}^{(t)}, \mathbf{y}^{(t)}, \delta', b_1, b_2, \eta, \mu$) then
        return $(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})$;
    end if
end for
return $(\mathbf{x}^{(T)}, \mathbf{y}^{(T)})$;
```

The Sinkhorn algorithm thus has a number of tunable parameters (precision, error parameters, iteration count). We show how to choose them in such a way that the resulting quantum algorithm obtains an ε-relative-entropy-scaling in time $\widetilde{O}(\sqrt{mn}/\varepsilon^2)$.

▶ **Theorem 4.** *Let $\mathbf{A} \in [0,1]^{n\times n}$ be a matrix with $\|\mathbf{A}\|_1 \leq 1$ and m non-zero entries, each rational and at least $\mu > 0$, let $\mathbf{r}, \mathbf{c} \in (0,1]^n$ with $\|\mathbf{r}\|_1 = \|\mathbf{c}\|_1 = 1$, and let $\varepsilon \in (0,1]$. Assume $\mathbf{A}$ is asymptotically scalable to $(\mathbf{r}, \mathbf{c})$. Then there exists a quantum algorithm that, given sparse oracle access to $\mathbf{A}$, with probability $\geq 2/3$, computes $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n$ such that $\mathbf{A}(\mathbf{x}, \mathbf{y})$ is ε-relative-entropy-scaled to $(\mathbf{r}, \mathbf{c})$, for a total time complexity of $\widetilde{O}(\sqrt{mn}/\varepsilon^2)$.*

A generalization of Pinsker's inequality (cf. [9, Lem. 2.1]) implies the following corollary.

▶ **Corollary 5.** *Let $\mathbf{A}, \mathbf{r}, \mathbf{c}$ and ε be as in Theorem 4. Then there exists a quantum algorithm that with probability $\geq 2/3$ computes $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n$ such that $\mathbf{A}(\mathbf{x}, \mathbf{y})$ is ε-ℓ_1-scaled to $(\mathbf{r}, \mathbf{c})$, for a total time complexity of $\widetilde{O}(\sqrt{mn}/\varepsilon^4)$.*

In [9, Thm. C.6], we show that if the matrix $\mathbf{A}$ is entrywise-positive, then the number of iterations to obtain an ε-relative-entropy-scaling can be reduced to roughly $1/\sqrt{\varepsilon}$ rather than roughly $1/\varepsilon$, leading to the following theorem.

Procedure ApproxScalingFactor$(\mathbf{a}, r, \mathbf{y}, \delta, b_1, b_2, \eta, \mu)$.

Input: Oracle access to rational $\mathbf{a} \in [0,1]^n$, rational $r \in (0,1]$, oracle access to $\mathbf{y} \in \mathbb{R}^n$ encoded in (b_1, b_2) fixed-point format, desired precision $\delta \in (0,1]$, desired failure prob. $\eta \in [0,1]$, lower bound $\mu > 0$ on non-zero entries of $\mathbf{a}$
Output: A number x encoded in (b_1, b_2) fixed-point format
Guarantee: If $b_1 \geq \lceil \log_2(|\ln(r/\sum_{j=1}^n a_j e^{y_j})|) \rceil$ and $b_2 \geq \lceil \log_2(1/\delta) \rceil$, then with prob. $\geq 1-\eta$, x is a δ-additive approximation of $\ln(r/\sum_{j=1}^n a_j e^{y_j})$

Procedure TestScaling$(\mathbf{A}, \mathbf{r}, \mathbf{c}, \mathbf{x}, \mathbf{y}, \delta, b_1, b_2, \eta, \mu)$.

Input: Oracle access to rational $\mathbf{A} \in [0,1]^{n\times n}$ with $\|\mathbf{A}\|_1 \leq 1$, rational $\mathbf{r}, \mathbf{c} \in (0,1]^n$, oracle access to $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ encoded in (b_1, b_2) fixed-point format, test precision $\delta \in (0,1)$, desired failure probability $\eta \in [0,1]$, lower bound $\mu > 0$ on non-zero entries of $\mathbf{A}$
Output: A bit indicating whether $\mathbf{x}, \mathbf{y}$ forms a δ-relative-entropy-scaling of $\mathbf{A}$ to target marginals $\mathbf{r}, \mathbf{c}$.
Guarantee: If $b_1 \geq \log_2(|\ln(r_\ell/\sum_{j=1}^n A_{\ell j} e^{y_j})|)$ for all $\ell \in [n]$, and similarly for the columns, and furthermore $b_2 \geq \lceil \log_2(1/\delta) \rceil$, then with probability at least $1-\eta$: outputs **False** if $D(\mathbf{r}\|\mathbf{r}(\mathbf{A}(\mathbf{x},\mathbf{y}))) \geq 2\delta$ or $D(\mathbf{c}\|\mathbf{c}(\mathbf{A}(\mathbf{x},\mathbf{y}))) \geq 2\delta$, outputs **True** if both are at most δ

▶ **Theorem 6.** *Let $\mathbf{A} \in [\mu, 1]^{n\times n}$ be a matrix with $\|\mathbf{A}\|_1 \leq 1$, each entry rational and at least $\mu > 0$, let $\mathbf{r}, \mathbf{c} \in (0,1]^n$ with $\|\mathbf{r}\|_1 = \|\mathbf{c}\|_1 = 1$, and let $\varepsilon \in (0,1]$. Then there exists a quantum algorithm that, given sparse oracle access to $\mathbf{A}$, with probability $\geq 2/3$, computes $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n$ such that $\mathbf{A}(\mathbf{x}, \mathbf{y})$ is ε-relative-entropy-scaled to $(\mathbf{r}, \mathbf{c})$, at a total time complexity of $\widetilde{O}(n^{1.5}/\varepsilon^{1.5})$.*

The next corollary also follows from the generalization of Pinsker's inequality.

▶ **Corollary 7.** *Let $\mathbf{A}, \mathbf{r}, \mathbf{c}$ and ε be as in Theorem 6. Then there exists a quantum algorithm that with probability $\geq 2/3$ computes $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n$ such that $\mathbf{A}(\mathbf{x}, \mathbf{y})$ is ε-ℓ_1-scaled to $(\mathbf{r}, \mathbf{c})$, for a total time complexity of $\widetilde{O}(\sqrt{mn}/\varepsilon^3)$.*

3.1 Potential argument

The analysis uses the convex potential function

$$f(\mathbf{x}, \mathbf{y}) = \sum_{i,j=1}^n A_{ij} e^{x_i + y_j} - \sum_{i=1}^n r_i x_i - \sum_{j=1}^n c_j y_j.$$

This function (already mentioned in the introduction) is often used in the context of matrix scaling, as its gradient is the difference between the current and desired marginals. Many of the more sophisticated algorithms for matrix scaling minimize this function directly. For our purposes, we first bound the potential gap $f(\mathbf{0}, \mathbf{0}) - \inf_{\mathbf{x},\mathbf{y} \in \mathbb{R}^n} f(\mathbf{x}, \mathbf{y})$ (see [9, App. A]).

▶ **Lemma 8** (Potential gap). *Assume $\mathbf{A} \in [0,1]^{n\times n}$ with $\|\mathbf{A}\|_1 \leq 1$ and non-zero entries at least $\mu > 0$. If $\mathbf{A}$ is asymptotically $(\mathbf{r}, \mathbf{c})$-scalable, then $f(\mathbf{0}, \mathbf{0}) - \inf_{\mathbf{x},\mathbf{y} \in \mathbb{R}^n} f(\mathbf{x}, \mathbf{y}) \leq \ln(1/\mu)$.*

For matrices $\mathbf{A}$ that are *exactly* $(\mathbf{r}, \mathbf{c})$-scalable, this bound is well-known (see e.g. [34, 20]), but to the best of our knowledge, it has not yet appeared in the literature when $\mathbf{A}$ is only assumed to be asymptotically scalable to $(\mathbf{r}, \mathbf{c})$.

One can show that, for a Sinkhorn iteration in which we update the rows exactly, i.e., $\hat{x}_\ell = \ln(r_\ell / \sum_{j=1}^n A_{\ell j} e^{y_j})$ for $\ell \in [n]$, the potential decreases by exactly the relative entropy:

$$f(\mathbf{x},\mathbf{y}) - f(\hat{\mathbf{x}},\mathbf{y}) = D(\mathbf{r}\|\mathbf{r}(\mathbf{A}(\mathbf{x},\mathbf{y}))), \tag{3.2}$$

and similarly for exact column updates. The next lemma generalizes this to allow for error in the update; it shows that we can lower bound the decrease of the potential function in every iteration in terms of the relative entropy between the target marginal and the current marginal, assuming every call to the subroutine `ApproxScalingFactor` succeeds.

▶ **Lemma 9.** *Let $\mathbf{A} \in \mathbb{R}_+^{n\times n}$, let $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, let $\delta \in [0,1]$, and let $\hat{\mathbf{x}} \in \mathbb{R}^n$ be a vector such that for every $\ell \in [n]$, we have $|\hat{x}_\ell - \ln(r_\ell / \sum_{j=1}^n A_{\ell j} e^{y_j})| \le \delta$. Then*

$$f(\mathbf{x},\mathbf{y}) - f(\hat{\mathbf{x}},\mathbf{y}) \ge D\big(\mathbf{r}\big\|\mathbf{r}(\mathbf{A}(\mathbf{x},\mathbf{y}))\big) - 2\delta.$$

A similar statement holds for an update of $\mathbf{y}$ (using $\mathbf{c}$ instead of $\mathbf{r}$ in the relative entropy).

When $\delta = 0$, the inequality becomes an equality which is well-known, see e.g. [2] or [20]. The lemma shows that updating the scaling vectors with additive precision δ suffices to make progress in minimizing the potential function f, as long as we are still $\Omega(\delta)$ away from the desired marginals (in relative entropy distance).

We thus wish to store the entries of $\mathbf{x}$ and $\mathbf{y}$ with additive precision $\delta > 0$. We want to do so using a (b_1, b_2) fixed-point format, so we need $b_2 \ge \lceil \log_2(1/\delta) \rceil$. The guarantees of `ApproxScalingFactor` and `TestScaling` assert that this choice of b_2 is also sufficient. Lemma 10 shows how large we need to take b_1 to ensure the requirements of `ApproxScalingFactor` and `TestScaling` are satisfied in every iteration. The algorithm returns as soon as `TestScaling` returns **True**, or after T iterations. However, to simplify the analysis, we always assume that $\mathbf{x}^{(t)}$ and $\mathbf{y}^{(t)}$ are defined for $t = 0,\dots,T$.

▶ **Lemma 10** (Bounding the scalings). *Let $\mathbf{A} \in [0,1]^{n\times n}$ with $\|\mathbf{A}\|_1 \le 1$ and non-zero entries at least $\mu > 0$. Let $T \ge 1$ and $\delta \in [0,1]$. Denote $\sigma = \max(|\ln r_{\min}|, |\ln c_{\min}|)$. Let $b_2 = \lceil \log_2(1/\delta) \rceil$ and choose $b_1 = \lceil \log_2(T) + \log_2(\ln(\frac{1}{\mu}) + 1 + \sigma) \rceil$. If for all $t \in [T]$ the subroutine `ApproxScalingFactor` succeeds, then for all $t \in [T]$ and $\ell \in [n]$ we have*

$$\left|\ln\left(\frac{r_\ell}{\sum_{j=1}^n A_{\ell j} e^{y_j^{(t)}}}\right)\right| \le 2^{b_1}, \quad \left|\ln\left(\frac{c_\ell}{\sum_{i=1}^n A_{i\ell} e^{x_i^{(t)}}}\right)\right| \le 2^{b_1}$$

and $\|(\mathbf{x}^{(t)},\mathbf{y}^{(t)})\|_\infty \le t\left(\ln\left(\frac{1}{\mu}\right) + \delta + \sigma\right) \le t\left(\ln\left(\frac{1}{\mu}\right) + 1 + \sigma\right)$.

To formally analyze the expected progress it is convenient to define the following events.

▶ **Definition 11** (Important events). *For $t = 1,\dots,T$, we define the following events:*

- *Let S_t be the event that all n calls to `ApproxScalingFactor` succeed in the t-th iteration.*
- *Define S to be the intersection of the events S_t, i.e., $S = \cap_{t=1}^T S_t$.*

To give some intuition, we note below that the event S is the "good" event where a row-update makes the relative entropy between $\mathbf{r}$ and the updated row-marginals at most δ (and similarly for the columns). We only use Lemma 12 in [9, App. C].

▶ **Lemma 12.** *If S holds and $\delta \le 1$, then the following holds for all $t \in [T]$:*

- *If t is odd, then $D(\mathbf{r}\|\mathbf{r}(\mathbf{A}(\mathbf{x}^{(t)},\mathbf{y}^{(t)}))) \le \delta$.*
- *If t is even, then $D(\mathbf{c}\|\mathbf{c}(\mathbf{A}(\mathbf{x}^{(t)},\mathbf{y}^{(t)}))) \le \delta$.*

We can combine Lemmas 8 and 9 to show Algorithm 1 returns, with high probability, an ε-relative-entropy-scaling to $(\mathbf{r}, \mathbf{c})$ by choosing $\delta = O(\varepsilon)$.

▶ **Proposition 13.** *Let $\mathbf{A} \in [0,1]^{n\times n}$ with $\|\mathbf{A}\|_1 \leq 1$ and non-zero entries at least $\mu > 0$ and rational, and let $\mathbf{r}, \mathbf{c} \in (0,1]^n$ with $\|\mathbf{r}\|_1 = \|\mathbf{c}\|_1 = 1$. Assume $\mathbf{A}$ is asymptotically scalable to $(\mathbf{r}, \mathbf{c})$. For $\varepsilon \in (0,1]$, choose $T = \left\lceil \frac{8}{\varepsilon} \ln\left(\frac{1}{\mu}\right)\right\rceil + 1$, $\delta = \frac{\varepsilon}{16}$, $\delta' = \frac{\varepsilon}{2}$, $\eta = \frac{1}{3(n+1)T}$, $b_2 = \lceil \log_2(\frac{1}{\delta})\rceil$, and $b_1 = \lceil \log_2(T) + \log_2(\ln(\frac{1}{\mu}) + \sigma + 1)\rceil$, where $\sigma = \max(|\ln r_{\min}|, |\ln c_{\min}|)$. Then, Algorithm 1 returns $(\mathbf{x}, \mathbf{y})$ s.t. $D(\mathbf{r}\|\mathbf{r}(\mathbf{A}(\mathbf{x},\mathbf{y}))) \leq \varepsilon$ and $D(\mathbf{c}\|\mathbf{c}(\mathbf{A}(\mathbf{x},\mathbf{y}))) \leq \varepsilon$ with probability $\geq 2/3$.*

3.2 Quantum approximate summing

`ApproxScalingFactor` and `TestScaling` both rely on the computation of additive approximations to numbers of the form $\ln(\sum_{i=1}^n a_i e^{y_i})$. Here we sketch our approach and mention some complications; see [9, Sec. 4] for details. If we assume the numbers $b_i = a_i e^{y_i}$ can be queried at unit cost, then we can efficiently compute $\ln(\sum_{i=1}^n b_i)$ up to additive error δ using amplitude estimation, as follows. After pre-processing (using *quantum maximum finding* [24]) one may assume that $b_i \in [0,1]$ and $\max_i b_i \geq 1/2$. With 2 queries to the b_is and a small number of other gates we prepare $\frac{1}{\sqrt{n}}\sum_{i=1}^n |i\rangle \left(\sqrt{b_i}\,|0\rangle + \sqrt{1-b_i}\,|1\rangle\right)$. The squared norm of the part ending in $|0\rangle$ equals $p = \frac{1}{n}\sum_{i=1}^n b_i \in [1/2n, 1]$. Let $\delta \in (0, 1/2]$ be an error parameter that we instantiate later. Using amplitude amplification we estimate p up to multiplicative error $1 \pm \delta$ using $O(\frac{1}{\delta}\sqrt{1/p}) = O(\frac{1}{\delta}\sqrt{n})$ queries to the b_is, and $\widetilde{O}(\frac{1}{\delta}\sqrt{n})$ gates, with error probability $\leq 1/3$ [15, Theorem 12]. We can reduce the error probability to a small $\eta > 0$, by running this $O(\log(1/\eta))$ times and outputting the median outcome. Naturally, multiplicative approximation of $\sum_{i=1}^n b_i$ yields additive approximation of $\ln(\sum_{i=1}^n b_i) = \ln(\sum_{i=1}^n a_i e^{y_i})$.

One obstacle to efficiently implementing the above is that one cannot simply compute all numbers to sufficient precision. For `ApproxScalingFactor` for instance, we aim to compute a number $\ln(r/\sum_{j=1}^n a_j e^{y_j})$ where the y_j can (and typically do) grow linearly with n, so we cannot compute e^{y_j} with sufficiently high precision in time sublinear in n. Instead we compute additive approximations of relative quantities such as $e^{y_i - y_j} \leq 1$ for $i, j \in [n]$ with $y_j \geq y_i$, and use properties of the log to relate this to the original desired quantity. This approach is widely used in practice, e.g., [4]. Note that these issues concern both the classical and quantum setting, but are particularly important for the latter, since we aim for a better dependence on m and n for the time complexity. We implement everything such that the fixed-point format (b_1, b_2) for both the input and output of the oracles is the same, avoiding the need to change the encoding format in every Sinkhorn or Osborne iteration.

3.3 Time complexity

Combining the above, we prove one of our main results (already stated earlier), bounding the time complexity of computing an ε-relative-entropy-scaling of $\mathbf{A}$ to marginals $(\mathbf{r}, \mathbf{c})$.

▶ **Theorem 4.** *Let $\mathbf{A} \in [0,1]^{n\times n}$ be a matrix with $\|\mathbf{A}\|_1 \leq 1$ and m non-zero entries, each rational and at least $\mu > 0$, let $\mathbf{r}, \mathbf{c} \in (0,1]^n$ with $\|\mathbf{r}\|_1 = \|\mathbf{c}\|_1 = 1$, and let $\varepsilon \in (0,1]$. Assume $\mathbf{A}$ is asymptotically scalable to $(\mathbf{r}, \mathbf{c})$. Then there exists a quantum algorithm that, given sparse oracle access to $\mathbf{A}$, with probability $\geq 2/3$, computes $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n$ such that $\mathbf{A}(\mathbf{x}, \mathbf{y})$ is ε-relative-entropy-scaled to $(\mathbf{r}, \mathbf{c})$, for a total time complexity of $\widetilde{O}(\sqrt{mn}/\varepsilon^2)$.*

Proof. We show that Algorithm 1 with the parameters chosen as in Proposition 13 has the stated time complexity. Note that the cost of computing these parameters from the input will be dominated by the runtime of the algorithm. Proposition 13 shows that Algorithm 1 runs for at most $O(\ln(1/\mu)/\varepsilon)$ iterations. Next we show the time complexity per iteration is $\widetilde{O}(\sqrt{mn}/\varepsilon)$, which implies the claimed total time complexity of $\widetilde{O}(\sqrt{mn}/\varepsilon^2)$.

Theorem 4.5 in [9] formalizes the discussion of Section 3.2: using `ApproxScalingFactor` with precision $\delta = \Theta(\varepsilon)$ on a row containing s potentially non-zero entries incurs a cost $\widetilde{O}(\sqrt{s}/\varepsilon)$, where we suppress a polylogarithmic dependence on n. One iteration of Algorithm 1 applies `ApproxScalingFactor` once to each row or once to each column, so by Cauchy–Schwarz the total cost of the calls to `ApproxScalingFactor` in one iteration is

$$\widetilde{O}\Big(\sum_{i=1}^{n} \sqrt{s_i^r}/\varepsilon + \sum_{j=1}^{n} \sqrt{s_j^c}/\varepsilon\Big) \subseteq \widetilde{O}(\sqrt{mn}/\varepsilon),$$

where we recall that s_i^r and s_j^c are the numbers of potentially non-zero entries in the i-th row and j-th column of $\mathbf{A}$, respectively, and m is the total number of potentially non-zero entries in $\mathbf{A}$ (i.e., $\sum_{i=1}^{n} s_i^r = m = \sum_{j=1}^{n} s_j^c$). Similarly, [9, Thm. 4.7] shows invoking `TestScaling` with precision $\delta' = \Theta(\varepsilon)$ incurs a cost of order $\widetilde{O}(\sqrt{mn}/\varepsilon)$. Finally we observe that compiling the quantum circuits (and preparing their inputs) for the calls to `ApproxScalingFactor` and `TestScaling` can be done with at most a polylogarithmic overhead. ◀

Note that the dependency on $\ln(1/\mu)$ is suppressed by the $\widetilde{O}$, since we assume the numerator and denominators of the rational inputs are bounded by a polynomial in n.

3.4 Complications in Osborne's algorithm

For the matrix balancing problem one can use a similar approach as for the matrix scaling problem. The idea is to fix the requirement of being ε-ℓ_1-balanced for individual coordinates, one at a time. More precisely, given an index $\ell \in [n]$, the update is given by $\mathbf{x}' = \mathbf{x} + \Delta_\ell \mathbf{e}_\ell$, where Δ_ℓ is chosen such that $r_\ell(\mathbf{A}(\mathbf{x}')) = c_\ell(\mathbf{A}(\mathbf{x}'))$. Expanding this and using $A_{\ell\ell} = 0$ yields $e^{\Delta_\ell} \cdot r_\ell(\mathbf{A}(\mathbf{x})) = e^{-\Delta_\ell} \cdot c_\ell(\mathbf{A}(\mathbf{x}))$. Since we assume every row and column contains at least one non-zero entry, the above equation has a unique solution, given by

$$\Delta_\ell = \ln\left(\sqrt{c_\ell(\mathbf{A}(\mathbf{x}))/r_\ell(\mathbf{A}(\mathbf{x}))}\right). \tag{3.3}$$

Note that the updates of multiple coordinates *cannot* be done simultaneously, since each coordinate can potentially affect all row and column marginals. This is in contrast with the Sinkhorn algorithm for matrix scaling, where all rows or all columns can be updated at the same time. This provides a significant challenge in the analysis of the algorithm since we can no longer test whether we have found an ε-balancing in between each iteration. We give a novel analysis of Osborne's algorithm [9, Sec. 6] and of a randomized version of Sinkhorn's algorithm [9, Sec. 5] that shows a uniformly random iterate provides an ε-balancing/scaling with high probability. For matrix balancing this yields the following.

▶ **Theorem 14.** *Let $\mathbf{A} \in [0,1]^{n\times n}$ be a matrix whose non-zero entries are rational, at least $\mu > 0$, with zeroes on the diagonal, each row and column having at least one non-zero element, and let $\varepsilon \in (0,1]$. Assume $\mathbf{A}$ is asymptotically balanceable. Then there exists a quantum algorithm that, given sparse oracle access to $\mathbf{A}$, returns with probability $\geq 2/3$ a vector $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{A}(\mathbf{x})$ is ε-ℓ_1-balanced, with expected time complexity $\widetilde{O}(\sqrt{mn}/\varepsilon^4)$.*

4 Matching lower bound for matrix scaling with constant ε

We show that our algorithm for matrix scaling is in fact optimal with respect to the dependence on n and m, for constant $\varepsilon > 0$. We prove an $\Omega(\sqrt{mn})$ lower bound on the query complexity of quantum algorithms for $\Theta(1)$-ℓ_1-scaling to the uniform marginals $(\mathbf{1}/n, \mathbf{1}/n)$. Here we sketch the case $m = n^2$; Section 7 in [9] gives the full proof also for the sparse case.

We consider the problem of learning a permutation "modulo two" in the following sense.[4]

▶ **Definition 15** (Single-bit descriptor). *Let $\sigma \in S_n$ be a permutation. The* single-bit descriptor *of σ is the bit string $z \in \{0,1\}^n$ with entries $z_i \equiv \sigma(i) \bmod 2$.*

We first use the adversary method [5] to prove an $\Omega(n\sqrt{n})$ bound for recovering the single-bit descriptor, given (dense) oracle access to the permutation matrix. This is tight, since one can use Grover on each column to fully recover the permutation matrix. We follow a similar proof structure as in the $\Omega(\sqrt{n})$-query lower bound given in [5] for finding $\sigma^{-1}(1)$, and the $\Omega(n\sqrt{n})$-query lower bound for graph connectivity [23].

▶ **Lemma 16.** *Let n be a positive multiple of 4. Given an $n \times n$ permutation matrix $\mathbf{P}$ corresponding to a permutation σ (i.e., $P_{ij} = \delta_{i,\sigma(j)}$ for $i, j \in [n]$), recovering the single-bit descriptor z of σ, with success probability at least $2/3$, requires $\Omega(n\sqrt{n})$ quantum queries to a dense matrix oracle for $\mathbf{P}$.*

One can then boost this lower bound to show that even learning a (certain) constant fraction of the entries of the single-bit descriptor of a permutation requires $\Omega(n\sqrt{n})$ quantum queries. (The precise constant depends on those in Lemma 16 and in Grover search.) We then reduce the problem of learning the single-bit descriptor to the scaling problem, by replacing each 1-entry of the permutation matrix by one of two 2×2 gadget matrices (and each 0-entry by the 2×2 all-0 matrix). These gadgets are such that we can determine (most of) the single-bit descriptor from the column-scaling vectors $\mathbf{y}$ of an $\Theta(1)$-ℓ_1-scaling to uniform marginals. Explicitly, the gadget matrices are as follows:

$$\mathbf{B}_0 = \begin{bmatrix} \frac{2}{9} & \frac{4}{9} \\ \frac{1}{9} & \frac{2}{9} \end{bmatrix}, \qquad \mathbf{B}_1 = \begin{bmatrix} \frac{4}{9} & \frac{2}{9} \\ \frac{2}{9} & \frac{1}{9} \end{bmatrix},$$

Note that the two matrices have the same columns, but in reverse order. We show in the next lemma that from an approximate scaling of $\mathbf{B}_i$ to uniform marginals, one can recover the bit i.

▶ **Lemma 17.** *The matrices $\mathbf{B}_0, \mathbf{B}_1 \in \{\frac{1}{9}, \frac{2}{9}, \frac{4}{9}\}^{2\times 2}$ are entrywise positive, with entries summing to one, and they are exactly scalable to uniform marginals. For $i \in \{0,1\}$, let $(\mathbf{x}, \mathbf{y})$ be $\frac{1}{8}$-ℓ_1-scaling vectors for $\mathbf{B}_i$ to uniform marginals. If $i = 0$ then $y_1 - y_2 > 0.18$, while if $i = 1$ then $y_1 - y_2 < -0.18$. Moreover, $(\mathbf{x}, (y_2, y_1))$ are $\frac{1}{8}$-ℓ_1-scaling vectors for $\mathbf{B}_{1-i}$ to uniform marginals.*

In other words, the matrices can be distinguished just by learning the column-scaling vectors, but they have the same set of possible row-scaling vectors.

[4] Alternatively, one could consider the problem of learning an entire permutation, which would simplify the notation and proofs slightly. However, for the reduction to matrix scaling, this seems to require gadget matrices of size roughly $\log_2 n \times \log_2 n$, leading to a slightly weaker lower bound.

Proof. Since one matrix is obtained by swapping the columns of the other, the last claim is immediately clear, and it suffices to prove the remaining claims for $\mathbf{B}_0$.

First, we note that $\mathbf{B}_0$ is exactly scalable, since

$$\begin{bmatrix} \frac{3}{4} & 0 \\ 0 & \frac{3}{2} \end{bmatrix} \begin{bmatrix} \frac{2}{9} & \frac{4}{9} \\ \frac{1}{9} & \frac{2}{9} \end{bmatrix} \begin{bmatrix} \frac{3}{2} & 0 \\ 0 & \frac{3}{4} \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

has uniform marginals. Now suppose that $(\mathbf{x}, \mathbf{y})$ is an $\frac{1}{8}$-ℓ_1-scaling of $\mathbf{B}_0$ to uniform marginals. By the requirement on the column marginals, we have

$$\left(\frac{2}{9}e^{x_1} + \frac{1}{9}e^{x_2}\right) e^{y_1} \geq \frac{1}{2} - \frac{1}{8} \quad \text{and} \quad \left(\frac{4}{9}e^{x_1} + \frac{2}{9}e^{x_2}\right) e^{y_2} \leq \frac{1}{2} + \frac{1}{8}.$$

By dividing the first inequality by the second one we get

$$\frac{1}{2} \cdot \frac{e^{y_1}}{e^{y_2}} \geq \frac{3}{5},$$

and so $y_1 - y_2 \geq \ln \frac{6}{5} > 0.18$. ◀

Together with Lemma 16 this leads to the following lower bound.

▶ **Theorem 18.** *There exists a constant $\varepsilon \in (0, 1)$ such that any quantum algorithm which, given a sparse oracle for an $n \times n$-matrix that is exactly scalable to uniform marginals and has m potentially non-zero entries which sum to 1, returns an ε-ℓ_1-scaling with probability $\geq 2/3$, requires $\Omega(\sqrt{mn})$ quantum queries to the oracle.*

References

1 Zeyuan Allen-Zhu, Yuanzhi Li, Rafael Oliveira, and Avi Wigderson. Much faster algorithms for matrix scaling. In *Proceedings of IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS'17)*, pages 890–901, 2017.

2 Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Advances in Neural Information Processing Systems*, volume 30, pages 1964–1974, 2017.

3 Jason M. Altschuler and Pablo A. Parrilo. Approximating Min-Mean-Cycle for low-diameter graphs in near-optimal time and memory, 2020. `arXiv:2004.03114`.

4 Jason M. Altschuler and Pablo A. Parrilo. Random Osborne: A simple, practical algorithm for matrix balancing in near-linear time, 2020. `arXiv:2004.02837`.

5 Andris Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002. Earlier version in STOC'00. quant-ph/0002066.

6 E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. SIAM, 1999. `doi:10.1137/1.9780898719604`.

7 Joran van Apeldoorn and András Gilyén. Improvements in quantum SDP-solving with applications. In *Proceedings of 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 99:1–99:15, 2019. `doi:10.4230/LIPIcs.ICALP.2019.99`.

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

9 Joran van Apeldoorn, Sander Gribling, Yinan Li, Harold Nieuwboer, Michael Walter, and Ronald de Wolf. Quantum algorithms for matrix scaling and matrix balancing, 2020. `arXiv:2011.12823v1`.

10 Srinivasan Arunachalam and Reevu Maity. Quantum boosting. In *Proceedings of 37th International Conference on Machine Learning (ICML'20)*, 2020. `arXiv:2002.05056`.

11 Boaz Barak, Zeev Dvir, Amir Yehudayoff, and Avi Wigderson. Rank bounds for design matrices with applications to combinatorial geometry and locally correctable codes. In *Proceedings of 43rd Symposium on Theory of Computing (STOC'11)*, pages 519–528. ACM, 2011.

12 Andrew Michael Bradley. *Algorithms for the equilibration of matrices and their application to limited-memory Quasi-Newton methods*. PhD thesis, Stanford University, 2010.

13 Fernando G. S. L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu. Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning. In *Proceedings of 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:14, 2019. `doi:10.4230/LIPIcs.ICALP.2019.27`.

14 Fernando G. S. L. Brandão and Krysta M. Svore. Quantum speed-ups for solving semidefinite programs. In *Proceedings of IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS'17)*, pages 415–426, 2017. `doi:10.1109/FOCS.2017.45`.

15 Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *Contemporary Mathematics*, pages 53–74. American Mathematical Society, 2002. `arXiv:quant-ph/0005055`.

16 Peter Bürgisser, Cole Franks, Ankit Garg, Rafael Oliveira, Michael Walter, and Avi Wigderson. Efficient algorithms for tensor scaling, quantum marginals, and moment polytopes. In *Proceedings of 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS'18)*, pages 883–897, 2018. `doi:10.1109/FOCS.2018.00088`.

17 Peter Bürgisser, Cole Franks, Ankit Garg, Rafael Oliveira, Michael Walter, and Avi Wigderson. Towards a theory of non-commutative optimization: geodesic 1st and 2nd order methods for moment maps and polytopes. In *Proceedings of 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS'19)*, pages 845–861. IEEE, 2019.

18 Peter Bürgisser, Ankit Garg, Rafael Oliveira, Michael Walter, and Avi Wigderson. Alternating minimization, scaling algorithms, and the null-cone problem from invariant theory. In *Proceedings of 9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:20, 2018. `doi:10.4230/LIPIcs.ITCS.2018.24`.

19 Peter Bürgisser, Yinan Li, Harold Nieuwboer, and Michael Walter. Interior-point methods for unconstrained geometric programming and scaling problems, 2020. `arXiv:2008.12110`.

20 Deeparnab Chakrabarty and Sanjeev Khanna. Better and simpler error analysis of the Sinkhorn–Knopp algorithm for matrix scaling. *Mathematical Programming*, pages 1–13, 2020.

21 Michael B. Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. Matrix scaling and balancing via box constrained Newton's method and interior point methods. In *Proceedings of IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS'17)*, pages 902–913, 2017. `doi:10.1109/FOCS.2017.88`.

22 Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, pages 2292–2300, 2013.

23 Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006. `doi:10.1137/050644719`.

24 Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum, 1996. `arXiv:quant-ph/9607014`.

25 Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. In *Proceedings of 16th Annual IEEE Conference on Computational Complexity*, pages 100–106, 2001. `doi:10.1109/CCC.2001.933877`.

26 Ankit Garg, Leonid Gurvits, Rafael Oliveira, and Avi Wigderson. Algorithmic and optimization aspects of Brascamp-Lieb inequalities, via operator scaling. *Geometric and Functional Analysis*, 28(1):100–145, 2018. Earlier version in STOC'17.

27 Ankit Garg, Leonid Gurvits, Rafael Oliveira, and Avi Wigderson. Operator scaling: theory and applications. *Foundations of Computational Mathematics*, pages 1–68, 2019. Earlier version in FOCS'16.

28 Ankit Garg and Rafael Oliveira. Recent progress on scaling algorithms and applications. *Bulletin of the EATCS, Computational Complexity Column*, 125, 2018. `arXiv:1808.09669`.

29 Leonid Gurvits. Classical complexity and quantum entanglement. *Journal of Computer and System Sciences*, 69(3):448–484, 2004. `doi:10.1016/j.jcss.2004.06.003`.

30 Yassine Hamoudi, Maharshi Ray, Patrick Rebentrost, Miklos Santha, Xin Wang, and Siyi Yang. Quantum algorithms for hedging and the Sparsitron, 2020. `arXiv:2002.06003`.

31 Martin Idel. A review of matrix scaling and Sinkhorn's normal form for matrices and positive maps, 2016. `arXiv:1609.06349`.

32 Adam Izdebski and Ronald de Wolf. Improved quantum boosting, 2020. `arXiv:2009.08360`.

33 B. Kalantari, L. Khachiyan, and A. Shokoufandeh. On the complexity of matrix balancing. *SIAM Journal on Matrix Analysis and Applications*, 18(2):450–463, 1997. `doi:10.1137/S0895479895289765`.

34 B. Kalantari, I. Lari, F. Ricca, and B. Simeone. On the complexity of general matrix scaling and entropy minimization via the RAS algorithm. *Mathematical Programming*, 112:371–401, 2008.

35 Bahman Kalantari and Leonid Khachiyan. On the rate of convergence of deterministic and randomized RAS matrix scaling algorithms. *Operations Research Letters*, 14(5):237–244, 1993. `doi:10.1016/0167-6377(93)90087-W`.

36 Bahman Kalantari and Leonid Khachiyan. On the complexity of nonnegative-matrix scaling. *Linear Algebra and its Applications*, 240:87–103, 1996. `doi:10.1016/0024-3795(94)00188-X`.

37 J. Kruithof. Telefoonverkeersrekening. *De Ingenieur*, 52:E15–E25, 1937.

38 Nathan Linial, Alex Samorodnitsky, and Avi Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Combinatorica*, 20(4):545–568, 2000. URL: `https://www.math.ias.edu/~avi/PUBLICATIONS/MYPAPERS/LSW98/lsw00.pdf`.

39 Oren E. Livne and Gene H. Golub. Scaling by binormalization. *Numerical Algorithms*, 35(1):97–120, 2004.

40 Mathworks. balance: diagonal scaling to improve eigenvalue accuracy. URL: `https://www.mathworks.com/help/matlab/ref/balance.html`.

41 Arkadi Nemirovski and Uriel Rothblum. On complexity of matrix scaling. *Linear Algebra and its Applications*, 302-303:435–460, 1999. `doi:10.1016/S0024-3795(99)00212-8`.

42 Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. A primal-dual operator splitting method for conic optimization. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016. `arXiv:1312.3039`.

43 E. E. Osborne. On pre-conditioning of matrices. *Journal of ACM*, 7(4), 1960. `doi:10.1145/321043.321048`.

44 Rafail Ostrovsky, Yuval Rabani, and Arman Yousefi. Matrix balancing in L_p norms: Bounding the convergence rate of Osborne's iteration. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, pages 154–169, 2017. `doi:10.1137/1.9781611974782.11`.

45 B. N. Parlett and C. Reinsch. Balancing a matrix for calculation of eigenvalues and eigenvectors. *Numerische Mathematik*, 13:293–304, 1969.

46 Thomas Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 1762–1769, 2011.

47 Uriel G. Rothblum and Hans Schneider. Scalings of matrices which have prespecified row sums and column sums via optimization. *Linear Algebra and its Applications*, 114:737–764, 1989.

48 Leonard J. Schulman and Alistair Sinclair. Analysis of a classical matrix preconditioning algorithm. In *Proceedings of 47th Annual ACM Symposium on Theory of Computing (STOC'15)*, pages 831–840, 2015.

49 Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

50 Richard Stone. *Multiple classifications in social accounting.* University of Cambridge, Department of Applied Economics, 1964.