J Optim Theory Appl (2017) 172:436–454
DOI 10.1007/s10957-016-0990-2

Stochastic Matrix-Free Equilibration

Steven Diamond[1] · **Stephen Boyd**[1]

Received: 21 February 2016 / Accepted: 27 July 2016 / Published online: 5 August 2016

Abstract We present a novel method for approximately equilibrating a matrix using only multiplication by the matrix and its transpose. Our method is based on convex optimization and projected stochastic gradient descent, using an unbiased estimate of a gradient obtained by a randomized method. Our method provably converges in expectation and empirically gets good results with a small number of iterations. We show how the method can be applied as a preconditioner for matrix-free iterative algorithms, substantially reducing the iterations required to reach a given level of precision. We also derive a novel connection between equilibration and condition number, showing that equilibration minimizes an upper bound on the condition number over all choices of row and column scalings.

Keywords Matrix balancing · Preconditioning · Convex optimization · Stochastic optimization

Mathematics Subject Classification 65F35

1 Introduction

Equilibration refers to scaling the rows and columns of a matrix so the norms of the rows are the same and the norms of the columns are the same. Equilibration has applications to a variety of problems, including target tracking in sensor networks [1], web page

✉ Steven Diamond
diamond@cs.stanford.edu

Stephen Boyd
boyd@stanford.edu

[1] Stanford University, Stanford, CA, USA

ranking [2], and adjusting contingency tables to match known marginal probabilities [3]. The primary use of equilibration, however, is as a heuristic method for reducing condition number [4]; in turn, reducing condition number is a heuristic for speeding up a variety of iterative algorithms [5, Chap. 5], [6,7]. Using equilibration to accelerate iterative algorithms is connected to the broader notion of diagonal preconditioning, which has been a subject of research for decades; see, e.g., [8, Chap. 2], [9, Chap. 10], [10,11].

Equilibration has several justifications as a heuristic for reducing condition number. We will show in Sect. 3.2 that, if a matrix is square and nonsingular, equilibrating the matrix minimizes a tight upper bound on the condition number over all possible row and column scalings. Scaling only the rows or only the columns of a matrix so they have the same norms (rather than scaling both at once, as we do here) also has a connection with minimizing condition number [12].

Equilibration is an old problem, and many techniques have been developed for it, such as the Sinkhorn–Knopp [13] and Ruiz algorithms [14]. Existing equilibration methods require knowledge of the entries of the matrix. For this reason, equilibration cannot be used in matrix-free methods, which only interact with the matrix via multiplication of a vector by the matrix or by its transpose. Such matrix-free methods play a crucial role in scientific computing and optimization. Examples include the conjugate gradients method [15], LSQR [16], and the Chambolle–Cremers–Pock algorithm [17].

In this paper, we introduce a stochastic matrix-free equilibration method that provably converges in expectation to the correct row and column scalings. Our method builds on work by Bradley [4], who proposed a matrix-free equilibration algorithm with promising empirical results but no theoretical guarantees. We sketch out our approach in Sect. 2 and develop it in detail in Sects. 3 and 4, including proving theoretical convergence rates for our algorithm. We present numerical examples in Sect. 5 that show our matrix-free equilibration method converges far more quickly than the theoretical analysis suggests, delivering effective equilibration in a few tens of iterations, each involving one multiplication by the matrix and one by its transpose. In Sect. 6, we apply our method to several matrix-free iterative algorithms. We observe that the cost of equilibration is more than compensated for by the speedup of the iterative algorithm due to diagonal preconditioning. In Sect. 7, we show how our method can be modified to handle variants of the equilibration problem, such as symmetric and block equilibration.

2 Problem and Approach

In this section, we formally define the equilibration problem and outline our approach to solving it. Given a matrix $A \in \mathbb{R}^{m\times n}$, equilibration means finding diagonal matrices $D \in \mathbb{R}^{m\times m}$ and $E \in \mathbb{R}^{n\times n}$ so that the rows of DAE all have ℓ_p-norm α and the columns of DAE all have ℓ_p-norm β. (The row and column norm values α and β are related by $m\alpha^p = n\beta^p$ for $p < \infty$.) Common choices of p are 1, 2, and ∞; in this paper, we will focus on ℓ_2-norm equilibration. Without loss of generality, we assume throughout that the entries of D and E are nonnegative.

As discussed in Sect. 1, many algorithms have been developed for ℓ_p-norm equilibration, but they all involve knowledge of (the entries of) the matrix $|A|^p$, where the function $|\cdot|^p$ is applied elementwise. Our contribution is the first theoretically grounded algorithm for equilibration in a matrix-free setting, in which the matrix A is a black box that can only be accessed via multiplication by A and by A^{T}.

Our approach is based on representing equilibration as a convex optimization problem, as in [18]. We add well-motivated regularization and constraints that make the objective function strongly convex and bound the Lipschitz constant. We develop a method for getting an unbiased estimate of the gradient of the objective function, which we use to solve the problem via projected stochastic gradient descent. Our algorithm converges in expectation to the optimal value of the problem with rate $O(1/t)$, where t is the number of multiplications by A and A^{T}.

3 Equilibration via Convex Optimization

3.1 Convex Formulation

Equilibration can be posed as the convex optimization problem [18]

$$\text{minimize} \left[(1/2) \sum_{i=1}^{m} \sum_{j=1}^{n} (A_{ij})^2 \mathrm{e}^{2u_i + 2v_j} - \alpha^2 \mathbf{1}^{\mathrm{T}} u - \beta^2 \mathbf{1}^{\mathrm{T}} v \right], \tag{1}$$

where $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$ are the optimization variables [19]. The notation $\mathbf{1}$ indicates a vector whose entries are all one. The diagonal matrices D and E are obtained via

$$D = \mathbf{diag}\left(\mathrm{e}^{u_1}, \ldots, \mathrm{e}^{u_m}\right), \qquad E = \mathbf{diag}\left(\mathrm{e}^{v_1}, \ldots, \mathrm{e}^{v_n}\right).$$

The optimality conditions for problem (1) are precisely that DAE is equilibrated, i.e.,

$$|DAE|^2 \mathbf{1} = \alpha^2 \mathbf{1}, \qquad \left|EA^{\mathrm{T}}D\right|^2 \mathbf{1} = \beta^2 \mathbf{1}.$$

Problem (1) is unbounded below precisely when the matrix A cannot be equilibrated.

Problem (1) can be solved using a variety of methods for smooth convex optimization [5,20]. One attractive method, which exploits the special structure of the objective, is to alternately minimize over u and v. We minimize over u (or equivalently D) by setting

$$D_{ii} = \alpha \left(\sum_{j=1}^{n} A_{ij}^2 E_{jj}^2 \right)^{-1/2}, \quad i = 1, \ldots, m.$$

We minimize over $v(E)$ by setting

$$E_{jj} = \beta \left(\sum_{i=1}^{m} A_{ij}^2 D_{ii}^2 \right)^{-1/2}, \quad j = 1, \ldots, n.$$

When $m = n$ and $\alpha = \beta = 1$, the above updates are precisely the Sinkhorn–Knopp algorithm. In other words, the Sinkhorn–Knopp algorithm is alternating block minimization for the problem (1).

3.2 Equilibration and Condition Number

In this subsection, we show that equilibrating a square matrix minimizes an upper bound on the condition number. We will not use these results in the sequel, where we focus on matrix-free methods for equilibration.

For $U \in \mathbb{R}^{n \times n}$ nonsingular define the function Φ by

$$\Phi(U) = \exp\left(\|U\|_F^2/2 \right) / \det\left(U^{\mathrm{T}} U \right)^{1/2} = \exp\left(\sum_{i=1}^{n} \sigma_i^2/2 \right) / \prod_{i=1}^{n} \sigma_i,$$

where $\sigma_1 \geq \cdots \geq \sigma_n > 0$ are the singular values of U. (Here $\|U\|_F$ denotes the Frobenius norm.)

Theorem 3.1 *Let A be square and invertible. Then, diagonal D and E equilibrate A, with row and column norms one, if and only if they minimize $\Phi(DAE)$ over D and E diagonal.*

Proof We first rewrite problem (1) in terms of D and E to obtain

$$\begin{array}{ll} \text{minimize } (1/2)\|DAE\|_F^2 - \displaystyle\sum_{i=1}^{n} \log D_{ii} - \sum_{j=1}^{n} \log E_{jj} \\ \text{subject to } \mathbf{diag}(D) > 0, \quad \mathbf{diag}(E) > 0, \quad D, E \text{ diagonal}, \end{array}$$

(Here we take $\alpha = \beta = 1$, so the row and column norms are one.) We can rewrite this problem as

$$\begin{array}{ll} \text{minimize } (1/2)\|DAE\|_F^2 - \log\det\left((DAE)^{\mathrm{T}} (DAE) \right)^{1/2} \\ \text{subject to } \mathbf{diag}(D) > 0, \quad \mathbf{diag}(E) > 0, \quad D, E \text{ diagonal}, \end{array}$$

since the objective differs from the problem above only by the constant term $\log\det\left(A^{\mathrm{T}} A\right)^{1/2}$. Finally, taking the exponential of the objective, we obtain the equivalent problem

$$\text{minimize } \Phi(DAE) = \exp\left((1/2)\|DAE\|_F^2\right) / \det\left((DAE)^{\mathrm{T}}(DAE)\right)^{1/2}$$
$$\text{subject to } \mathbf{diag}(D) > 0, \quad \mathbf{diag}(E) > 0, \quad D, E \text{ diagonal.}$$

Thus, diagonal (positive) D and E equilibrate A, with row and column norms one, if and only if they minimize the objective of this problem. □

Theorem 3.1 links equilibration with minimization of condition number because the function $\Phi(DAE)$ gives an upper bound on $\kappa(DAE)$.

Theorem 3.2 *Let $U \in \mathbb{R}^{n\times n}$ have singular values $\sigma_1 \geq \cdots \geq \sigma_n > 0$ and condition number $\kappa = \sigma_1/\sigma_n$. Then,*

$$2e^{-n/2}\Phi(U) \geq \kappa. \tag{2}$$

Moreover, this inequality is tight within a factor of 2, i.e., there exists U with condition number κ with

$$2e^{-n/2}\Phi(U) \leq 2\kappa. \tag{3}$$

Proof We factor Φ into

$$\Phi(U) = \Psi(\sigma_1, \sigma_n)\prod_{i=2}^{n-1}\Gamma(\sigma_i),$$

where

$$\Psi(\sigma_1, \sigma_n) = \exp\left(\left(\sigma_1^2 + \sigma_n^2\right)/2\right)/(\sigma_1\sigma_n), \qquad \Gamma(\sigma_i) = \exp\left(\sigma_i^2/2\right)/\sigma_i.$$

We first relate Ψ and the condition number, by minimizing $\Psi(\sigma_1, \sigma_n)$ with $\sigma_1 = \kappa\sigma_n$ (i.e., with condition number κ). We must minimize over σ_n the function

$$\Psi(\kappa\sigma_n, \sigma_n) = \frac{\exp\left(\sigma_n^2\left(1+\kappa^2\right)/2\right)}{\kappa\sigma_n^2}.$$

With change of variable $z = \sigma_n^2$, this function is convex. The minimizer is $z = 2/(1+\kappa^2)$ and the minimum value $(e/2)(\kappa + 1/\kappa)$. Therefore, we have

$$\Psi(\sigma_1, \sigma_n) \geq (e/2)(\kappa + 1/\kappa).$$

It is straightforward to show that $\Gamma(\sigma_i)$ is convex and minimized when $\sigma_i = 1$. Thus, we have $\Gamma(\sigma_i) \geq \Gamma(1) = \mathrm{e}^{1/2}$. We combine these results to obtain the inequality

$$\Phi(U) \geq \left(\mathrm{e}^{n/2}/2\right)(\kappa + 1/\kappa), \tag{4}$$

which is sharp; indeed, it is tight when

$$\sigma_1 = \left(\frac{2\kappa^2}{1+\kappa^2}\right)^{1/2}, \qquad \sigma_n = \left(\frac{2}{1+\kappa^2}\right)^{1/2},$$

and $\sigma_i = 1$ for $i = 2, \ldots, n-1$.

The inequality (4) implies inequality (2), since $\kappa + 1/\kappa \geq \kappa$. With the values of σ_i that make (4) tight, the inequality (3) holds because $\kappa + 1/\kappa \leq 2\kappa$. □

Theorems 3.1 and 3.2 show that equilibration is the same as minimizing $\Phi(DAE)$ over diagonal D and E and that $\Phi(DAE)$ is an upper bound on $\kappa(DAE)$, the condition number of DAE.

3.3 Regularized Equilibration

The equilibration problem and its equivalent convex optimization problem (1) suffer from several flaws. The first is that not all matrices can be equilibrated [4]. For example, if the nonzero matrix A has a zero row or column, it cannot be equilibrated. As a less obvious example, a triangular matrix with unit diagonal cannot be equilibrated. When the matrix A cannot be equilibrated, the convex problem (1) is unbounded [18].

The second flaw is that even when the matrix A can be equilibrated problem (1) does not have a unique solution. Given a solution $(u^\star, v^\star)$ to problem (1), the point $(u^\star + \gamma, v^\star - \gamma)$ is a solution for any $\gamma \in \mathbb{R}$. In other words, we can scale D by e^{γ} and E by $\mathrm{e}^{-\gamma}$ and still have DAE equilibrated. We would prefer to guarantee a solution where D and E have roughly the same scale. The final flaw is that in practice we do not want the entries of D and E to be extremely large or extremely small; we may have limits on how much we are willing to scale the rows or columns.

We address these flaws by modifying the problem (1), adding regularization and box constraints, and reframing the equilibration problem as

$$\begin{aligned} \text{minimize } (1/2) \sum_{i=1}^{m} \sum_{j=1}^{n} (A_{ij})^2 \mathrm{e}^{2u_i + 2v_j} - \alpha^2 \mathbf{1}^{\mathrm{T}} u - \beta^2 \mathbf{1}^{\mathrm{T}} v \\ + (\gamma/2) \left(\|u\|_2^2 + \|v\|_2^2 \right) \\ \text{subject to } \|u\|_\infty \leq M, \quad \|v\|_\infty \leq M, \end{aligned} \tag{5}$$

where $\gamma > 0$ is the regularization parameter and the parameter $M > 0$ bounds the entries of D and E to lie in the interval $\left[\mathrm{e}^{-M}, \mathrm{e}^{M}\right]$. The additional regularization term penalizes large choices of u and v (which correspond to large or small row and column scalings). It also makes the objective strictly convex and bounded below, so the modified problem (5) always has a unique solution $(u^\star, v^\star)$, even when A cannot be equilibrated. Assuming that the constraints are not active at the solution, we have

$$\mathbf{1}^{\mathrm{T}} u^\star = \mathbf{1}^{\mathrm{T}} v^\star,$$

which means that the optimal D and E have the same scale in the sense that the products of their diagonal entries are equal:

$$\prod_{i=1}^{m} D_{ii} = \prod_{j=1}^{n} E_{jj}.$$

Problem (5) is convex and can be solved using a variety of methods. Block alternating minimization over u and v can be used here, as in the Sinkhorn–Knopp algorithm. We minimize over u (or equivalently D) by setting

$$D_{ii} = \Pi_{[e^{-2M},e^{2M}]}\left(2\alpha^2/\gamma - W\left(2e^{2\alpha^2/\gamma}\sum_{j=1}^{n} A_{ij}^2 E_{jj}^2/\gamma\right)\right)^{1/2},$$

for $i = 1, \ldots, m$. Here W is the Lambert W function [21] and $\Pi_{[e^{-2M},e^{2M}]}$ denotes projection onto the interval $[e^{-2M}, e^{2M}]$. We minimize over v (E) by setting

$$E_{jj} = \Pi_{[e^{-2M},e^{2M}]}\left(2\beta^2/\gamma - W\left(2e^{2\beta^2/\gamma}\sum_{i=1}^{m} A_{ij}^2 D_{ii}^2/\gamma\right)\right)^{1/2},$$

for $j = 1, \ldots, n$. When $M = +\infty$, $m = n$, and $\alpha = \beta = 1$, the D and E updates converge to the Sinkhorn–Knopp updates as $\gamma \to 0$ [22]. This method works very well, but like the Sinkhorn–Knopp method requires access to the individual entries of A, and so is not appropriate as a matrix-free algorithm.

Of course, solving problem (5) does not equilibrate A exactly; unless $\gamma = 0$ and the constraints are not active, its optimality conditions are not that DAE is equilibrated. We can make the equilibration more precise by decreasing the regularization parameter γ and increasing the scaling bound M. But if we are using equilibration as a heuristic for reducing condition number, approximate equilibration is more than sufficient.

4 Stochastic Method

In this section, we develop a method for solving problem (5) that is matrix-free, i.e., only accesses the matrix A by multiplying a vector by A or by A^T. (Of course we can find all the entries of A by multiplying A by the unit vectors e_i, $i = 1, \ldots, n$; then, we can use the block minimization method described above to solve the problem. But our hope is to solve the problem with far fewer multiplications by A or A^T.)

4.1 Unbiased Gradient Estimate

Gradient expression Let $f(u, v)$ denote the objective function of problem (5). The gradient $\nabla_u f(u, v)$ is given by

$$\nabla_u f(u, v) = |DAE|^2\mathbf{1} - \alpha^2 + \gamma u.$$

Similarly, the gradient $\nabla_v f(u, v)$ is given by

$$\nabla_v f(u, v) = \left|EA^T D\right|^2\mathbf{1} - \beta^2 + \gamma v.$$

The first terms in these expressions, $|DAE|^2\mathbf{1}$ and $\left|EA^{\mathrm{T}}D\right|^2\mathbf{1}$, are the row norms squared of the matrices DAE and $EA^{\mathrm{T}}D$, respectively. These are readily computed if we have access to the entries of A, but in a matrix-free setting, where we can only access A by multiplying a vector by A or A^{T}, it is difficult to evaluate these row norms. Instead, we will estimate them using a randomized method.

Estimating row norms squared Given a matrix $B \in \mathbb{R}^{m\times n}$, we use the following approach to get an unbiased estimate z of the row norms squared $|B|^2\mathbf{1}$. We first sample a random vector $s \in \mathbb{R}^n$ whose entries $s_i \in \{-1, 1\}$ are drawn independently with identical distribution (IID), with probability one half for each outcome. We then set $z = |Bs|^2$. This technique is discussed in [4,23,24].

To see that $E[z] = |B|^2\mathbf{1}$, consider $\left(b^{\mathrm{T}}s\right)^2$, where $b \in \mathbb{R}^n$. The expectation of $\left(b^{\mathrm{T}}s\right)^2$ is given by

$$E\left[\left(b^{\mathrm{T}}s\right)^2\right] = \sum_{i=1}^{n} b_i^2 E\left[s_i^2\right] + \sum_{i\neq j} b_i b_j E[s_i s_j] = \sum_{i=1}^{n} b_i^2.$$

As long as the entries of s are IID with mean 0 and variance 1, we have $E\left[\left(b^{\mathrm{T}}s\right)^2\right] = \sum_{i=1}^{n} b_i^2$. Drawing the entries of s from $\{-1, 1\}$, however, minimizes the variance of $\left(b^{\mathrm{T}}s\right)^2$.

4.2 Projected Stochastic Gradient

Method We follow the projected stochastic gradient method described in [25] and [26, Chap. 6], which solves convex optimization problems of the form

$$\text{minimize } f(x) \quad \text{subject to } x \in C, \tag{6}$$

where $x \in \mathbb{R}^n$ is the optimization variable, $f : \mathbb{R}^n \to \mathbb{R}$ is a strongly convex differentiable function, and C is a convex set, using only an oracle that gives an unbiased estimate of ∇f, and projection onto C.

We cannot evaluate $f(x)$ or $\nabla f(x)$, but we can evaluate a function $g(x, \omega)$ and sample from a distribution Ω such that $E_{\omega\sim\Omega}\, g(x, \omega) = \nabla f(x)$. Let μ be the strong convexity constant for f and $\Pi_C : \mathbb{R}^n \to \mathbb{R}^n$ denote the Euclidean projection onto C. Then, the method consists of T iterations of the update

$$x^t := \Pi_C\left(x^{t-1} - \eta_t g(x^{t-1}, \omega)\right),$$

where $\eta_t = 2/(\mu(t + 1))$ and ω is sampled from Ω. The final approximate solution $\bar{x}$ is given by the weighted average

$$\bar{x} = \sum_{t=0}^{\mathrm{T}} \frac{2(t + 1)}{(T + 1)(T + 2)} x^t.$$

Algorithm (1) gives the full projected stochastic gradient method in the context of problem (5). Recall that the objective of problem (5) is strongly convex with strong convexity parameter γ.

Algorithm 1 Projected stochastic gradient method for problem (5).

Input: $u^0 = 0$, $v^0 = 0$, $\bar{u} = 0$, $\bar{v} = 0$, and $\alpha, \beta, \gamma, M > 0$.

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

$D \leftarrow \mathbf{diag}\left(\mathrm{e}^{u_1^{t-1}}, \ldots, \mathrm{e}^{u_m^{t-1}}\right)$, $\quad E \leftarrow \mathbf{diag}\left(\mathrm{e}^{v_1^{t1}}, \ldots, \mathrm{e}^{v_n^{t-1}}\right)$.

Draw entries of $s \in \mathbb{R}^n$ and $w \in \mathbb{R}^m$ IID uniform from $\{-1, 1\}$.

$u^t \leftarrow \Pi_{[-M,M]^m}\left(u^{t-1} - 2\left(|DAEs|^2 - \alpha^2\mathbf{1} + \gamma u^{t-1}\right)/(\gamma(t+1))\right)$.

$v^t \leftarrow \Pi_{[-M,M]^n}\left(v^{t-1} - 2\left(\left|EA^{\mathrm{T}}Dw\right|^2 - \beta^2\mathbf{1} + \gamma v^{t-1}\right)/(\gamma(t+1))\right)$.

$\bar{u} \leftarrow 2u^t/(t+2) + t\bar{u}/(t+2)$.

$\bar{v} \leftarrow 2v^t/(t+2) + t\bar{v}/(t+2)$.

Output: $D = \mathbf{diag}\left(\mathrm{e}^{\bar{u}_1}, \ldots, \mathrm{e}^{\bar{u}_m}\right)$ and $E = \mathbf{diag}\left(\mathrm{e}^{\bar{v}_1}, \ldots, \mathrm{e}^{\bar{v}_n}\right)$.

Convergence rate Algorithm (1) converges in expectation to the optimal value of problem (5) with rate $O(1/t)$ [25]. Let $f(u, v) : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ denote the objective of problem (5), let $(u^\star, v^\star)$ denote the problem solution, and let $\tilde{g}(u, v, s, w) : \mathbb{R}^m \times \mathbb{R}^n \times \{-1, 1\}^n \times \{-1, 1\}^m \to \mathbb{R}^{m+n}$ be the estimate of $\nabla f(u, v)$ given by

$$\tilde{g}(u, v, s, w) = \begin{bmatrix} |DAEs|^2 - \alpha^2\mathbf{1} + \gamma u \\ \left|EA^{\mathrm{T}}Dw\right|^2 - \beta^2\mathbf{1} + \gamma v \end{bmatrix}.$$

Then, after T iterations of the algorithm, we have

$$E_{(u^{\mathrm{T}}, v^{\mathrm{T}}), \ldots, (u^1, v^1)} f(\bar{u}, \bar{v}) - f(u^\star, v^\star) \leq \frac{C}{\mu(T+1)},$$

where C is a constant bounded above by

$$C \leq \max_{(u,v) \in [-M,M]^{m \times n}} 2E_{s,w} \|\tilde{g}(u, v, s, w)\|_2^2.$$

In the expectation, s and w are random variables with entries drawn IID uniform from $\{-1, 1\}$.

We can make the bound more explicit. It is straightforward to show the equality

$$\begin{aligned} E_{s,w} \|\tilde{g}(u, v, s, w)\|_2^2 = \|\nabla f(u, v)\|_2^2 &- 4\mathbf{1}^{\mathrm{T}} |DAE|^4 \mathbf{1} \\ &+ 3\mathbf{1}^{\mathrm{T}} \left||DAE|^2 \mathbf{1}\right|^2 + 3\mathbf{1}^{\mathrm{T}} \left|\left|EA^{\mathrm{T}}D\right|^2 \mathbf{1}\right|^2, \end{aligned}$$

whereas before the function $|\cdot|^p$ is applied elementwise. The inequality

$$\max_{(u,v)\in[-M,M]^{m\times n}} \|\nabla f(u,v)\|_2^2 \leq \|\nabla f(M1, M1)\|_2^2 + 4\gamma M(\alpha^2 m + \beta^2 n),$$

is equally straightforward. We combine these two results to obtain the bound

$$C/2 \leq \|\nabla f(M1, M1)\|_2^2 + 4\gamma M(\alpha^2 m + \beta^2 n) + \mathrm{e}^{8M}\left(3\mathbf{1}^{\mathrm{T}}\left||A|^2\mathbf{1}\right|^2 + 3\mathbf{1}^{\mathrm{T}}\left|\left|A^{\mathrm{T}}\right|^2\mathbf{1}\right|^2 - 4\mathbf{1}^{\mathrm{T}}|A|^4\mathbf{1}\right).$$

Our bound on C is quite large. A more thorough analysis could improve the bound by considering the relative sizes of the different parameters and entries of A. For instance, it is straightforward to show that for $t = 1, \ldots, T$ we have

$$u_i^t \leq \alpha^2/\gamma, \quad i = 1, \ldots, m, \qquad v_j^t \leq \beta^2/\gamma, \quad j = 1, \ldots, n,$$

which gives a tighter bound if $\alpha^2/\gamma < M$ or $\beta^2/\gamma < M$. In any case, we find that in practice no more than tens of iterations are required to reach an approximate solution.

5 Numerical Experiments

We evaluated algorithm (1) on many different matrices A. We only describe the results for a single numerical experiment, but we obtained similar results for our other experiments. For our numerical experiment, we generated a sparse matrix $\hat{A} \in \mathbb{R}^{m\times n}$, with $m = 2 \times 10^4$ and $n = 10^4$, with 1% of the entries chosen uniformly at random to be nonzero, and nonzero entries drawn IID from a standard normal distribution. We next generated vectors $\hat{u} \in \mathbb{R}^m$ and $\hat{v} \in \mathbb{R}^n$ with entries drawn IID from a normal distribution with mean 1 and variance 1. We set the final matrix A to be

$$A = \mathbf{diag}\left(\mathrm{e}^{\hat{u}_1}, \ldots, \mathrm{e}^{\hat{u}_m}\right) \hat{A} \mathbf{diag}\left(\mathrm{e}^{\hat{v}_1}, \ldots, \mathrm{e}^{\hat{v}_n}\right).$$

We ran algorithm (1) for 1000 iterations to obtain an approximate solution $f(\bar{u}, \bar{v})$. We used the parameters $\alpha = (n/m)^{1/4}$, $\beta = (m/n)^{1/4}$, $\gamma = 10^{-1}$, and $M = \log(10^4)$. We obtained the exact solution $p^\star$ to high accuracy using Newton's method with backtracking line search. (Newton's method does not account for constraints, but we verified that the constraints were not active at the solution.)

Figure 1 plots the relative optimality gap $(f(\bar{u}, \bar{v}) - p^\star)/f(0, 0)$ and the RMS equilibration error,

$$\frac{1}{\sqrt{m+n}}\left(\sum_{i=1}^m \left(\sqrt{e_i^{\mathrm{T}}|DAE|^2\mathbf{1}} - \alpha\right)^2 + \sum_{j=1}^n \left(\sqrt{e_j^{\mathrm{T}}\left|EA^{\mathrm{T}}D\right|^2\mathbf{1}} - \beta\right)^2\right)^{1/2},$$

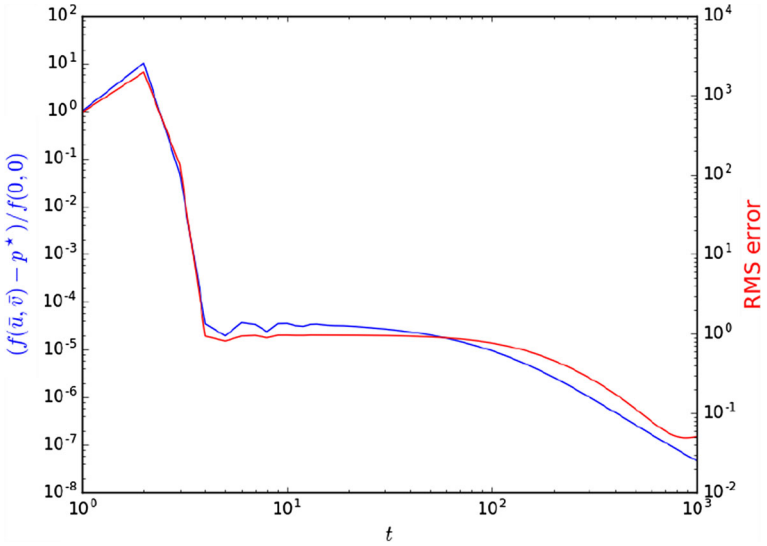

Fig. 1 Problem (5) optimality gap and RMS error versus iterations t

versus iteration. The RMS error shows how close DAE is to equilibrated; we do not expect it to converge to zero because of the regularization.

The objective value and RMS error decrease quickly for the first few iterations, with oscillations, and then decrease smoothly but more slowly. The slopes of the lines show the convergence rate. The least-squares linear fit for the optimality gap has slope -2.0, which indicates that the convergence was (much) faster than the theoretical upper bound $1/t$.

Figure 2 shows the condition number of DAE versus iteration. While equilibration merely minimizes an upper bound on the condition number, in this case the condition number corresponded quite closely with the objective of problem (5). The plot shows that after 4 iterations $\kappa(DAE)$ is back to the original condition number $\kappa(A) = 10^4$. After 100 iterations, the condition number is reduced by $200\times$, and it continues to decrease with further iterations.

6 Applications

Diagonal preconditioning can accelerate the convergence of many iterative matrix-free algorithms for solving linear equations or optimization problems. Our stochastic matrix-free equilibration method can be used with any of these algorithms as a general-purpose tool for generating a diagonal preconditioner. Generating a preconditioner through equilibration requires additional matrix multiplications, but ideally the algorithm's improved convergence rate due to the preconditioner will reduce the number of matrix multiplications needed overall. In this section, we investigate the effects of preconditioning using our equilibration method through numerical examples. In the first example, we solve a linear system using the preconditioned LSQR algorithm. In

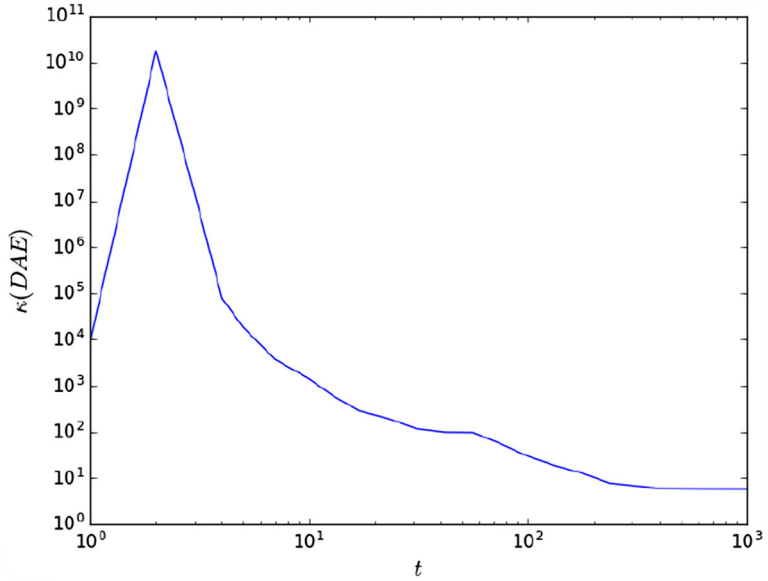

Fig. 2 Condition number of DAE versus iterations t

the second example, we solve a Lasso problem using the preconditioned Chambolle–Cremers–Pock algorithm. We find in both examples that preconditioning using our equilibration method substantially reduces the total number of matrix multiplications needed to reach a given level of precision. We expect that preconditioning via matrix-free equilibration would yield similar improvements for many other algorithms.

6.1 LSQR

The LSQR algorithm [16] is an iterative matrix-free method for solving the linear system $Ax = b$, where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, and $b \in \mathbb{R}^n$. Each iteration of LSQR involves one multiplication by A and one by A^{T}. LSQR is equivalent in exact arithmetic to applying the conjugate gradients method [15] to the normal equations $A^{\mathrm{T}}Ax = A^{\mathrm{T}}b$, but in practice has better numerical properties. An upper bound on the number of iterations needed by LSQR to achieve a given accuracy grows with $\kappa(A)$ [5, Chap. 5]. Thus, decreasing the condition number of A via equilibration can accelerate the convergence of LSQR. (Since LSQR is equivalent to conjugate gradients applied to the normal equations, it computes the exact solution in n iterations, at least in exact arithmetic. But with numerical roundoff error this does not occur.)

We use equilibration as a preconditioner by solving the linear system $(DAE)\bar{x} = Db$ with LSQR instead of $Ax = b$; we then recover x from $\bar{x}$ via $x = E\bar{x}$. We measure the accuracy of an approximate solution $\bar{x}$ by the residual $\|Ax - b\|_2$ rather than by residual $\|DAE\bar{x} - Db\|_2$ of the preconditioned system, since our goal is to solve the original system $Ax = b$.

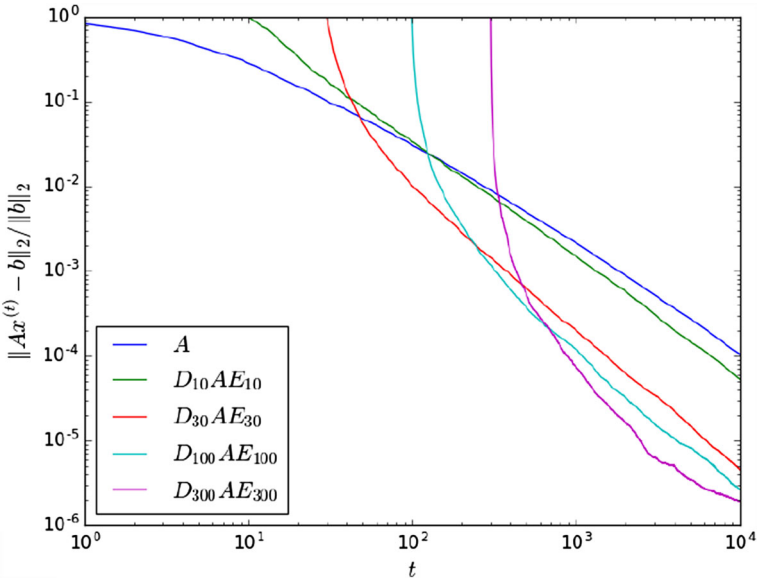

Fig. 3 Residual versus iterations t for LSQR

We compared the convergence rate of LSQR with and without equilibration. We generated the matrix $\hat{A} \in \mathbb{R}^{n\times n}$ as in Sect. 5, with $n = 10^4$. We chose $b \in \mathbb{R}^n$ by first generating $x^\star \in \mathbb{R}^n$ by drawing entries IID from a standard normal distribution and then setting $b = Ax^\star$.

We generated equilibrated matrices $D_{10}AE_{10}$, $D_{30}AE_{30}$, $D_{100}AE_{100}$, and $D_{300}AE_{300}$ by running algorithm (1) for 10, 30, 100, and 300 iterations, respectively. We used the parameters $\alpha = (n/m)^{1/4}$, $\beta = (m/n)^{1/4}$, $\gamma = 10^{-1}$, and $M = \log(10^4)$. Note that the cost of equilibration iterations is the same as the cost of LSQR iterations, since each involves one multiply by A and one by A^{T}.

Figure 3 shows the results of running LSQR with and without equilibration, from the initial iterate $x^0 = 0$. We show the relative residual $\|Ax^t - b\|_2/\|b\|_2$ versus iterations, counting the equilibration iterations, which can be seen as the original flat portions at the beginning of each curve. We can see that to achieve relative accuracy 10^{-4}, LSQR without preconditioning requires around 10^4 iterations; with preconditioning with 30 or more iterations of equilibration, it requires more than $10\times$ fewer iterations. We can see that higher accuracy justifies more equilibration iterations, but that the choice of just 30 equilibration iterations does very well. We can see that 10 iterations of equilibration are too few and only improve LSQR convergence a small amount.

6.2 Chambolle–Cremers–Pock

The Chambolle–Cremers–Pock (CCP) algorithm [17,27] is an iterative method for solving convex optimization problems of the form

$$\text{minimize } [f(x) + g(Ax)],$$

where $x \in \mathbb{R}^n$ is the variable, $A \in \mathbb{R}^{m \times n}$ is problem data, and f and g are convex functions. Each iteration of CCP requires one multiplication by A and one by A^{T}. Chambolle and Pock do not show a dependence on $\kappa(A)$ in their analysis of the algorithm convergence rate, but we nonetheless might expect that equilibration will accelerate convergence.

We compared the convergence rate of CCP with and without equilibration on the Lasso problem [28, Sect. 3.4]

$$\text{minimize} \left[\|Ax - b\|_2^2/\sqrt{\lambda} + \sqrt{\lambda}\|x\|_1 \right].$$

We generated the matrix $A \in \mathbb{R}^{m \times n}$ as in Sect. 5, with dimensions $m = 10^4$ and $n = 2 \times 10^4$. We generated $b \in \mathbb{R}^m$ by first generating $\hat{x} \in \mathbb{R}^n$ by choosing $n/10$ entries uniformly at random to be nonzero and drawing those entries IID from a standard normal distribution. We then set $b = A\hat{x} + \nu$, where the entries of $\nu \in \mathbb{R}^m$ were drawn IID from a standard normal distribution. We set $\lambda = 10^{-3} \left\| A^{\mathrm{T}} b \right\|_\infty$ and found the optimal value $p^\star$ for the Lasso problem using CVXPY [29] and GUROBI [30].

We generated equilibrated matrices $D_{10}AE_{10}$, $D_{30}AE_{30}$, $D_{100}AE_{100}$, and $D_{300}AE_{300}$ by running algorithm (1) for 10, 30, 100, and 300 iterations, respectively. We used the parameters $\alpha = (n/m)^{1/4}$, $\beta = (m/n)^{1/4}$, $\gamma = 10^{-1}$, and $M = \log(10^4)$.

Figure 4 shows the results of running CCP with and without equilibration. We used the parameters $\tau = \sigma = 0.8/\|D_k A E_k\|_2$ and $\theta = 1$ and set all initial iterates to 0. We computed $\|D_k A E_k\|_2$ using the implicitly restarted Arnoldi method [31]. We show the relative optimality gap $(f(x^t) - p^\star)/f(0)$ versus iterations, counting the equilibration iterations, which can be seen as the original flat portions at the beginning of each curve. We can see that to achieve relative accuracy 10^{-6}, CCP without preconditioning requires around 1000 iterations; with preconditioning with 100 iterations of equilibration, it requires more than $4\times$ fewer iterations. CCP converges to a highly accurate solution with just 100 equilibration iterations, so additional equilibration iterations are unnecessary. We can see that 10 and 30 iterations of equilibration are too few and do not improve CCP's convergence.

7 Variants

In this section, we discuss several variants of the equilibration problem that can also be solved in a matrix-free manner.

Symmetric equilibration When equilibrating a symmetric matrix $A \in \mathbb{R}^{n \times n}$, we often want the equilibrated matrix DAE also to be symmetric. For example, to use equilibration as a preconditioner for the conjugate gradients method, DAE must be symmetric [15]. We make DAE symmetric by setting $D = E$.

Symmetric equilibration can be posed as the convex optimization problem

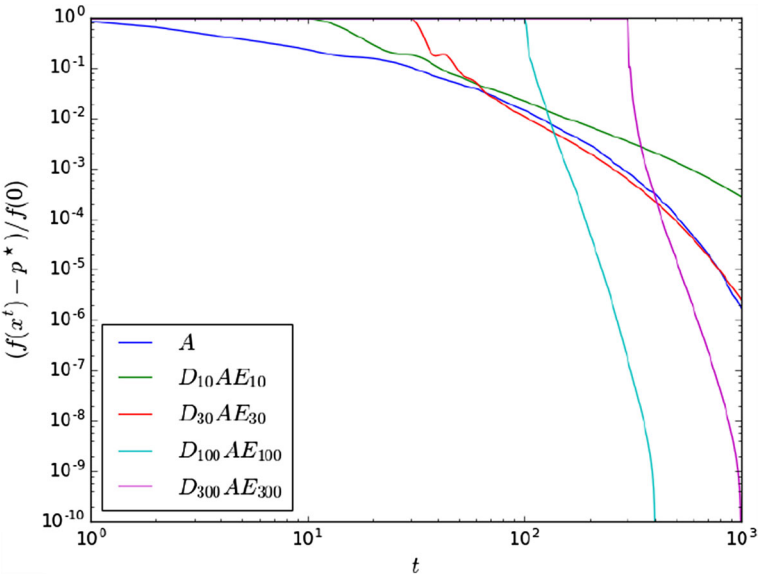

Fig. 4 Optimality gap versus iterations t for CCP

$$\text{minimize} \left[(1/4) \sum_{i=1}^{n} \sum_{j=1}^{n} (A_{ij})^2 \mathrm{e}^{2u_i + 2u_j} - \alpha^2 \mathbf{1}^{\mathrm{T}} u \right], \tag{7}$$

where $u \in \mathbb{R}^n$ is the optimization variable and $\alpha > 0$ is the desired value of the row and column norms. We approximately solve problem (7) by adding regularization and box constraints as in problem (5) and then applying algorithm (2), a simple modification of algorithm (1) with the same convergence guarantees.

Algorithm 2 Projected stochastic gradient method for symmetric equilibration.

Input: $u^0 = 0$, $\bar{u} = 0$, and α, γ, $M > 0$.

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

$D \leftarrow \mathbf{diag}\left(\mathrm{e}^{u_1^{t-1}}, \ldots, \mathrm{e}^{u_n^{t-1}}\right)$.

Draw entries of $s \in \mathbb{R}^n$ IID uniform from $\{-1, 1\}$.

$u^t \leftarrow \Pi_{[-M,M]^n}\left(u^{t-1} - 2\left(|DADs|^2 - \alpha^2 1 + \gamma u^{t-1}\right)/(\gamma(t+1))\right)$.

$\bar{u} \leftarrow 2u^t/(t+2) + t\bar{u}/(t+2)$.

Output: $D = \mathbf{diag}\left(\mathrm{e}^{\bar{u}_1}, \ldots, \mathrm{e}^{\bar{u}_n}\right)$.

Varying row and column norms In standard equilibration, we want all the row norms of DAE to be the same and all the column norms to be the same. We might instead want

the row and column norms to equal known vectors $r \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$, respectively. The vectors must satisfy $r^{\mathrm{T}} r = c^{\mathrm{T}} c$.

Equilibration with varying row and column norms can be posed as the convex optimization problem

$$\text{minimize} \left[(1/2) \sum_{i=1}^{m} \sum_{j=1}^{n} (A_{ij})^2 \mathrm{e}^{2u_i + 2v_j} - r^{\mathrm{T}} u - c^{\mathrm{T}} v \right], \tag{8}$$

where as usual $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$ are the optimization variables. We approximately solve problem (8) by adding regularization and box constraints as in problem (5) and then applying algorithm (1) with the appropriate modification to the gradient estimate.

Block equilibration A common constraint when using equilibration as a preconditioner is that the diagonal entries of D and E are divided into blocks that all must have the same value. For example, suppose we have a cone program

$$\text{minimize } c^{\mathrm{T}} x \quad \text{subject to } Ax + b \in \mathcal{K},$$

where $x \in \mathbb{R}^n$ is the optimization variable, $c \in \mathbb{R}^n, b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$ are problem data, and $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_\ell$ is a product of convex cones.

If we equilibrate A, we must ensure that $D\mathcal{K} = \mathcal{K}$. Let m_i be the dimension of cone $\mathcal{K}_i$. A simple sufficient condition for $D\mathcal{K} = \mathcal{K}$ is that D have the form

$$D = \mathbf{diag}\left(\mathrm{e}^{u_1} I_{m_1}, \ldots, \mathrm{e}^{u_p} I_{m_p}\right), \tag{9}$$

where $u \in \mathbb{R}^p$ and I_{m_i} is the m_i-by-m_i identity matrix. Given the constraint on D, we cannot ensure that each row of DAE has norm α. Instead, we view each block of m_i rows as a single vector and require that the vector have norm $\sqrt{m_i}\alpha$.

In the full block equilibration problem, we also require that E have the form

$$E = \mathbf{diag}\left(\mathrm{e}^{v_1} I_{n_1}, \ldots, \mathrm{e}^{v_q} I_{n_q}\right), \tag{10}$$

where $v \in \mathbb{R}^q$ and I_{n_j} is the n_j-by-n_j identity matrix. Again, we view each block of n_j columns as a single vector and require that the vector have norm $\sqrt{n_j}\beta$.

Block equilibration can be posed as the convex optimization problem

$$\text{minimize} \left[(1/2)\mathbf{1}^{\mathrm{T}} |DAE|^2 \mathbf{1} - \alpha^2 \sum_{i=1}^{p} u_i m_i - \beta^2 \sum_{j=1}^{q} v_j n_j \right], \tag{11}$$

where D and E are defined as in equations (9) and (10). We approximately solve problem (11) by adding regularization and box constraints as in problem (5) and then applying algorithm (1) with the appropriate modification to the gradient estimate. Our stochastic matrix-free block equilibration method is used in the matrix-free versions of the cone solvers SCS [32] and POGS [18] described in [33,34].

Tensor equilibration We describe here the case of three tensors; the generalization to higher-order tensors is clear. We are given a 3-dimensional array $A \in \mathbb{R}^{m\times n\times p}$ and seek coordinate scalings $d \in \mathbb{R}^m$, $e \in \mathbb{R}^n$, $f \in \mathbb{R}^p$ for which

$$\begin{aligned}
\left(\sum_{j=1}^{n}\sum_{k=1}^{p} A_{ijk}^2 d_i^2 e_j^2 f_k^2\right)^{1/2} &= \alpha, \quad i = 1,\ldots,m\\
\left(\sum_{i=1}^{m}\sum_{k=1}^{p} A_{ijk}^2 d_i^2 e_j^2 f_k^2\right)^{1/2} &= \beta, \quad j = 1,\ldots,n\\
\left(\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ijk}^2 d_i^2 e_j^2 f_k^2\right)^{1/2} &= \gamma, \quad k = 1,\ldots,p.
\end{aligned}$$

Here $\alpha, \beta, \gamma > 0$ are constants that satisfy $m\alpha^2 = n\beta^2 = p\gamma^2$.

Tensor equilibration can be posed as the convex optimization problem

$$\text{minimize} \left[(1/2)\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p}(A_{ijk}^2)\mathrm{e}^{2(u_i+v_j+w_k)} - \mathbf{1}^{\mathrm{T}}\begin{bmatrix}\alpha^2 u\\ \beta^2 v\\ \gamma^2 w\end{bmatrix}\right], \tag{12}$$

where $u \in \mathbb{R}^m$, $v \in \mathbb{R}^n$, and $w \in \mathbb{R}^p$ are the optimization variables. We can solve problem (12) using a simple variant of algorithm (1) that only interacts with the array A via the matrix-to-vector operations

$$\begin{aligned}
X &\to \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ijk} X_{ij}\\
Y &\to \sum_{i=1}^{m}\sum_{k=1}^{p} A_{ijk} Y_{ik}\\
Z &\to \sum_{j=1}^{n}\sum_{k=1}^{p} A_{ijk} Z_{jk}.
\end{aligned}$$

8 Conclusions

In this paper, by representing equilibration as a convex optimization problem, we derived a stochastic matrix-free equilibration method that provably converges in expectation with an $O(1/t)$ convergence rate. We showed in numerical experiments that our method converges in practice to an approximate solution in a few tens of iterations and that it provides an effective preconditioner for matrix-free iterative algorithms such as LSQR and Chambolle–Cremers–Pock. We also developed a novel interpretation of equilibration as minimizing an upper bound on condition number.

Acknowledgements The authors thank Reza Takapoui for helpful comments and pointers. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-114747 and by the DARPA XDATA and SIMPLEX programs.

References

1. Hwang, I., Balakrishnan, H., Roy, K., Shin, J., Guibas, L., Tomlin, C.: Multiple-target tracking and identity management. In: Proceedings of IEEE Sensors, pp. 36–41 (2003)
2. Knight, P.: The Sinkhorn–Knopp algorithm: convergence and applications. SIAM J. Matrix Anal. Appl. **30**(1), 261–275 (2008)
3. Schneider, M., Zenios, S.: A comparative study of algorithms for matrix balancing. Oper. Res. **38**(3), 439–455 (1990)
4. Bradley, A.: Algorithms for the equilibration of matrices and their application to limited-memory Quasi-Newton methods. Ph.D. thesis, Stanford University (2010)
5. Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York (2000)
6. Takapoui, R., Javadi, H.: Preconditioning via diagonal scaling. EE364b: Convex Optimization II Class Project (2014). http://stanford.edu/class/ee364b/projects/2014projects/reports/takapoui_javadi_report.pdf
7. Giselsson, P., Boyd, S.: Diagonal scaling in Douglas–Rachford splitting and ADMM. In: Proceedings of the IEEE Conference on Decision and Control, pp. 5033–5039 (2014)
8. Kelley, C.: Iterative Methods for Linear and Nonlinear Equations. Society for Industrial and Applied Mathematics, Philadelphia (1995)
9. Greenbaum, A.: Iterative Methods for Solving Linear Systems. Society for Industrial and Applied Mathematics, Philadelphia (1997)
10. Pock, T., Chambolle, A.: Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1762–1769 (2011)
11. Giselsson, P., Boyd, S.: Metric selection in fast dual forward-backward splitting. Automatica **62**, 1–10 (2015)
12. Sluis, A.: Condition numbers and equilibration of matrices. Numer. Math. **14**(1), 14–23 (1969)
13. Sinkhorn, R., Knopp, P.: Concerning nonnegative matrices and doubly stochastic matrices. Pac. J. Math. **21**(2), 343–348 (1967)
14. Ruiz, D.: A scaling algorithm to equilibrate both rows and columns norms in matrices. Technical report, Rutherford Appleton Lab., Oxon, UK, RAL-TR-2001-034 (2001)
15. Hestenes, M., Stiefel, E.: Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand. **49**(6), 409–436 (1952)
16. Paige, C., Saunders, M.: LSQR: an algorithm for sparse linear equations and sparse least squares. ACM Trans. Math. Softw. **8**(1), 43–71 (1982)
17. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. J. Math. Imaging Vis. **40**(1), 120–145 (2011)
18. Fougner, C., Boyd, S.: Parameter selection and pre-conditioning for a graph form solver. Preprint (2015). arXiv:1503.08366v1
19. Balakrishnan, H., Hwang, I., Tomlin, C.: Polynomial approximation algorithms for belief matrix maintenance in identity management. In: Proceedings of the IEEE Conference on Decision and Control, pp. 4874–4879 (2004)
20. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
21. Corless, R., Gonnet, G., Hare, D., Jeffrey, D., Knuth, D.: On the Lambert W function. Adv. Comput. Math. **5**(1), 329–359 (1996)
22. Hoorfar, A., Hassani, M.: Inequalities on the Lambert W function and hyperpower function. J. Inequal. Pure Appl. Math. **9**, 1–5 (2008)
23. Bekas, C., Kokiopoulou, E., Saad, Y.: An estimator for the diagonal of a matrix. Appl. Numer. Math. **57**(11–12), 1214–1229 (2007)
24. Hutchinson, M.: A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. Commun. Stat. Simul. Comput. **19**(2), 433–450 (1990)
25. Lacoste-Julien, S., Schmidt, M., Bach, F.: A simpler approach to obtaining an $O(1/t)$ convergence rate for the projected stochastic subgradient method. Preprint (2002). arXiv:1212.2002v2

26. Bubeck, S.: Convex optimization: algorithms and complexity. Found. Trends Mach. Learn. **8**(3–4), 231–357 (2015)
27. Pock, T., Cremers, D., Bischof, H., Chambolle, A.: An algorithm for minimizing the Mumford–Shah functional. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1133–1140 (2009)
28. Friedman, J., Hastie, T., Tibshirani, R.: The Elements of Statistical Learning, Springer Series in Statistics, vol. 1. Springer, New York (2001)
29. Diamond, S., Boyd, S.: CVXPY: a Python-embedded modeling language for convex optimization. J. Mach. Learn. Res. **17**(83), 1–5 (2016)
30. Gurobi Optimization, Inc.: Gurobi optimizer reference manual (2015). http://www.gurobi.com
31. Lehoucq, R., Sorensen, D.: Deflation techniques for an implicitly restarted Arnoldi iteration. SIAM J. Matrix Anal. Appl. **17**(4), 789–821 (1996)
32. O'Donoghue, B., Chu, E., Parikh, N., Boyd, S.: Conic optimization via operator splitting and homogeneous self-dual embedding. J. Optim. Theory Appl. **169**(3), 1042–1068 (2016)
33. Diamond, S., Boyd, S.: Convex optimization with abstract linear operators. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 675–683 (2015)
34. Diamond, S., Boyd, S.: Matrix-free convex optimization modeling. In: Goldengorin, B. (ed.) Optimization and Applications in Control and Data Sciences, Springer Optimization and Its Applications, vol. 115. Springer (2016, to appear)