OPTIMAL DIAGONAL PRECONDITIONING BEYOND WORST-CASE CONDITIONING: THEORY AND PRACTICE OF OMEGA SCALING *

SAEED GHADIMI†, WOOSUK L. JUNG‡, ARNESH SUJANANI‡, DAVID TORREGROSA-BELÉN§, AND HENRY WOLKOWICZ‡

Key words and phrases: κ, ω-condition numbers, diagonal preconditioning, iterative methods, PCG, linear systems, eigenvalues

Abstract. Preconditioning is essential in many areas of mathematics, and in particular is a fundamental tool for accelerating iterative methods for solving linear systems. In this work, we study optimal diagonal preconditioning under two distinct notions of conditioning: the classical worst-case κ-condition number and the averaging-based ω-condition number. We observe that ω-optimal preconditioning generally outperforms κ-optimal preconditioning.

For the κ-optimal preconditioning problem, we derive an affine-based pseudoconvex reformulation with three key advantages: all stationary points are global minima, subgradients are inexpensive to compute, and the optimization variable is an n-dimensional vector rather than an $n \times n$ matrix as in semidefinite programming (SDP) approaches. We develop a simple and highly efficient subgradient method, with convergence guarantees, for solving this pseudoconvex formulation. Numerical results indicate that our approach is substantially more scalable, efficient, and accurate than existing SDP-based methods, often achieving dramatic speedups.

For the ω-condition number, we provide explicit characterizations of optimal diagonal and block diagonal preconditioners. In particular, we show that several classical preconditioners, including Jacobi and row/column normalization, are ω-optimal, and that matrix balancing schemes monotonically reduce ω and converge to stationary points of the two-sided problem. To the best of our knowledge, this is the first unified and explicit characterization of optimality conditions for both κ and ω-based preconditioning.

Our numerical experiments further reveal a striking phenomenon: although κ-optimal preconditioners achieve stronger reductions in the worst-case condition number, ω-optimal preconditioners are substantially cheaper to compute and yield better performance for iterative methods such as preconditioned conjugate gradient (PCG) and least squares method (LSQR). Moreover, applying ω-optimal scaling to linear systems that are already κ-optimally preconditioned leads to further improvements in PCG iterations. These results suggest that the ω-condition number is more predictive of practical solver performance and highlight the advantages of ω-based preconditioning in large-scale settings.

1. Introduction.

Preconditioning plays a central role in accelerating iterative methods for large-scale linear systems, e.g., [10–12, 25]. All of these works focus on studying the classical κ-condition number of a matrix A (the ratio of largest to smallest singular values). On the other hand, another line of research has focused on studying the ω-condition number, which is the ratio of the arithmetic and geometric mean of singular values, [7, 17]. Although κ-optimal preconditioners minimize the worst-case condition number, our results show that ω-optimal preconditioners are cheaper to compute and are more predictive of PCG/LSQR performance.

Our work is divided into three parts. (i): First, motivated by the recent work in [12, 25] that evaluates κ-optimal diagonal preconditioners using convex semidefinite programming relaxations, we provide a different approach based on using the nonconvex formulation with κ. In particular, we provide a modified model that exploits the invariance of eigenvalues for a product of matrices and solves an essentially unconstrained pseudoconvex minimization problem. Our approach is significantly more efficient and robust, often achieving speedups of over 100x compared with the methods in [12, 25]. (ii): Second, we consider properties of the ω-condition number. In particular, we illustrate how to obtain explicit formulae for ω-optimal diagonal and block-diagonal preconditioners. In both settings, we obtain either known or new preconditioners, including the Jacobi preconditioner, column and row normalization preconditioners, and those obtained via matrix balancing algorithms such as Sinkhorn–Knopp. Compared to κ-optimal preconditioners, the ω-optimal counterparts are substantially cheaper to compute while retaining strong practical effectiveness. (iii): Finally, in our computational experiments we demonstrate the robustness and efficiency of our algorithms for finding κ-optimal preconditioners. We also demonstrate major advantages of using ω-optimal preconditioners for iteratively solving linear systems using preconditioned conjugate gradient (PCG) for positive definite systems and least squares method (LSQR) for general invertible systems. In general, we find that reductions in the ω-condition number are more strongly correlated with improvements in LSQR and PCG iteration counts

*EMAILS RESP.: SGHADIMI@UWATERLOO.CA, W2JUNG@UWATERLOO.CA, A3SUJANA@UWATERLOO.CA, DAVID.TORREGROSA@UA.ES, HWOLKOWICZ@UWATERLOO.CA

†Department of Management Science and Engineering, University of Waterloo, ON, Canada

‡ Department of Combinatorics and Optimization, University of Waterloo, ON, Canada

§ Department of Mathematics, University of Alicante, Alicante, Spain

than reductions in the κ-condition number.

For simplicity, we restrict our comparisons to diagonal and block diagonal preconditioning for large scale sparse positive definite and general invertible linear systems. The positive definite case arises from many diverse applications including: finite element analysis, sparse regression, Newton type optimization algorithms, and also from e.g., in [12,13]: (i) the so-called normal equations from interior point methods in solving linear programs and (ii) the Hessians in minimizing logistic regression.

The κ-condition number and ω-condition number for the positive definite case $M \succ 0$ are, respectively, given by ratios of eigenvalues

$$\kappa(M) = \frac{\lambda_{\max}(M)}{\lambda_{\min}(M)}; \quad \omega(M) = \frac{\mathrm{tr}(M)/n}{\det(M)^{1/n}} = \frac{\sum_i \lambda_i(M)/n}{\prod_i (\lambda_i(M))^{1/n}}. \tag{1.1}$$

For our purposes we extend these definitions in (1.1) using eigenvalues, rather than singular values, to nonsymmetric matrices with real positive eigenvalues allowing for simplified models for minimizing κ. Indeed, κ and ω can be considered as *worst-case* and *average-case* condition numbers, respectively. In fact, it is known that the ratio of the arithmetic to geometric mean is approximately the *coefficient of variation,* $\boldsymbol{CV}$; namely, the ratio between the standard deviation and the mean of a distribution; when this quantity is close to zero.[1] And, one of the big advantages of ω over κ is that finding explicit formulae for optimal preconditioners by minimizing ω with special structure is often possible due to the analyticity and simplicity of differentiation of trace and determinant.

1.1. Main Contributions. The main contributions of this paper consist of both theoretical and numerical analyses of the κ and ω-condition numbers. First, we provide an affine based pseudoconvex reformulation of the κ-optimal diagonal preconditioning problem that allows for an elegant characterization of its optimality conditions. There are three advantages of this reformulation: all its stationary points are global minima, its subgradients are inexpensive to compute, and its optimization variable is just an $n \times 1$ vector rather than a $n \times n$ matrix as in *semidefinite programming, SDP* [25]. Our extensive numerical experiments show that subgradient methods based on our reformulation can effectively converge to a κ-optimal diagonal preconditioner more efficiently in time and accuracy than current SDP-based approaches in the literature [12,25].

Second, we provide and exploit the ability to find explicit formulae for ω-optimal diagonal and block diagonal preconditioners. We demonstrate that many popular classic preconditioners, such as the Jacobi preconditioner and row/column normalization preconditioners are ω-optimal preconditioners. We also display that matrix balancing schemes reduce ω at every iteration and converge to a stationary point of the two-sided ω-optimal diagonal preconditioning problem. To the best of our knowledge, this is the first time that such a comprehensive characterization of optimality conditions for both condition numbers is provided.

Finally, we conduct additional extensive numerical experiments that compare the efficiency of the κ- and ω-condition numbers. Our results show that since ω-optimal preconditioners typically have closed-form solutions, they are much cheaper to construct than κ-optimal preconditioners. Our results further demonstrate that PCG iterations and LSQR iterations for solving linear systems are, on average, more correlated with the ω-condition number than the κ-condition number.

We now continue with the organization of the paper and more details on the main contributions. In Section 2, we present several formulations for minimizing κ along with the derivatives and optimality conditions. The formulations are presented to take advantage of the affine approach and then avoid the positive homogeneity of the problem in order to improve stability of the problem. In particular, we include Theorem 2.7 that presents a characterization of a κ-optimal diagonally preconditioned A that is based on the *largest/smallest* eigenpairs. This leads to an efficient subgradient algorithm and we include convergence results. In Section 3, we characterize ω-optimal preconditioners with special structures. A characterization for a ω-optimal diagonal preconditioner is simple as it corresponds to the classical Jacobi preconditioner for symmetric positive definite matrices and the row/column normalization preconditioner for general nonsingular matrices. We also show that the matrix balancing algorithm, Sinkhorn–Knopp, linearly converges to a stationary point of the two-sided ω-optimal diagonal preconditioning problem. Finally, we include results on extending from diagonal to block diagonal preconditioning in Subsection 3.3. We show

[1] The Young–Trent formula [34] establishes $GM/AM \approx 1 - \frac{1}{2}\mathbf{CV}^2$. Moreover, Taylor's approximation of $1/(1+x)$ around zero yields $AM/GM \approx 1 + \frac{1}{2}\mathbf{CV}^2$.

that the right-sided ω-optimal block-diagonal preconditioner for tall full-rank matrices A comes from taking Q-less QR decompositions of the blocks of A. We also give a characterization for the left-sided ω-optimal block-diagonal preconditioner and show that when A is square that it is related to applying the right-sided optimal preconditioner to A^T.

Extensive numerical tests are then conducted in Section 4. We demonstrate that our subgradient methods are more scalable and efficient at computing κ-optimal diagonal preconditioners than existing SDP-based approaches in the literature [12,25]. Moreover, the resulting preconditioners yield more substantial improvements in PCG performance for solving linear systems compared to those produced by [12,25].

We further compare optimal diagonal preconditioning strategies based on the condition numbers κ and ω. In particular, we observe that the Sinkhorn–Knopp algorithm, which converges to a stationary point of the two-sided ω-optimal preconditioning problem, produces preconditioners that significantly outperform the two-sided κ-optimal method of [25] in terms of LSQR iteration counts for minimizing $\|b - Ax\|$. While Sinkhorn–Knopp dramatically reduces ω, the method of [25] achieves a stronger reduction in κ. This is notable as despite this difference, Sinkhorn–Knopp yields better LSQR performance on the majority of instances while also being computationally cheaper.

Finally, we demonstrate that applying ω-optimal diagonal scaling to positive definite linear systems that are already κ-optimally scaled leads to significant improvements in PCG performance. Overall, our results highlight two key advantages of ω-optimal preconditioners: they are inexpensive to compute and substantially reduce the iteration counts of iterative methods.

1.2. Background; Preliminaries. Given a linear system $Ax = b$ with $A \in \mathcal{M}^n, n \times n$ matrix, nonsingular, and $A \approx P_1 P_2$, preconditioning effectively solves the given system by solving the following system for y:

$$P_1^{-1} A P_2^{-1} y = P_1^{-1} b \text{ for } y, \quad x = P_2^{-1} y. \tag{1.2}$$

It is assumed that the solutions with P_1, P_2 are inexpensive.[2] Surveys are given in e.g., [1,5,26,32] and the references therein. There is no matrix-matrix multiplication in preconditioning algorithms, such as PCG, as they do not form the products explicitly; and only matrix-vector multiplications/divisions are performed. For example, the well known *Jacobi preconditioner* uses the diagonal $d = \text{diag}(A)$, if it is not zero, and the diagonal matrices $P_2 = \text{Diag}(d), P_1 = I$.

A discussion and references on the two condition numbers κ, ω is given recently in [17]. The matrix nearness problem $\min_P \|I - AP\|_F$ is often used to find preconditioners P. This also arises in the derivation of quasi-Newton methods. The ω-condition number is introduced in [7] as a measure for nearness to the identity using $\min_{P>0} \omega(MP)$ for finding optimal quasi-Newton updates; see also [8,33,35]. It is related to the measure $\text{tr}\, A - \log\det A$ used in the convergence proofs in [3,4]. Then, Kaporin [18] used ω to derive new conjugate gradient convergence rate estimates and guarantees. More recently [2] presents further relationships and convergence analyses. As mentioned above, it is known that the ratio of AM to GM is approximately the *coefficient of variation*, $\boldsymbol{CV}$, $AM/GM \approx 1 + \frac{1}{2}CV^2$ [34]. When the data is log-normal, then we get the exact relation $AM/GM = \sqrt{1 + CV^2}$. Thus by reducing CV, ω promotes clustering of eigenvalues/singular values which is essential in convergence in PCG type methods, e.g., [14,22].

Note that for invertible $A \in \mathcal{M}^n$, [25] uses the former in $\sqrt{\kappa(A^T A)} = \kappa(A)$ as both involve largest to smallest singular values. In fact, as emphasized in [17], $\kappa(A) = \kappa(A^{-1})$ and it is the latter that is the operation for solving linear systems. However, these statements are are not true for ω, as the numerator being the sum of singular values, the nuclear norm, is not equivalent to $\sqrt{\text{tr}(A^T A)}$ and ω is not inverse invariant. In this paper, for ω, we work with general invertible A and then use $\omega(A^T A)$, i.e., use the ratios of eigenvalues of $A^T A$.

For certain structures in A, in contrast to κ, one can exploit the simple derivatives of $\text{tr}, \det$ in ω and obtain explicit formulae for the optimal preconditioner that in fact coincide with heuristics in the literature, see [17]. Thus no optimization algorithm is needed to obtain the optimal preconditioner. For example, if we restrict to a diagonal preconditioner $D = \text{diag}(d)$, then the classic Jacobi preconditioner is a multiple of the ω-optimal diagonal preconditioner. If we restrict to a diagonal structure along with a partial upper triangular part size k, then the diagonal part again corresponds to the Jacobi preconditioner, while the upper triangular part $D_{ij}, i \leqslant j \leqslant k$, yields exactly the formula for the Cholesky factorization. (see Proposition 3.10

[2] In some of our theoretical work for notational convenience, we use $D \leftarrow P^{-1}$, below.

for more details). Both these observations highlight the close connections ω-optimal preconditioners have with classical heuristic preconditioners.

1.2.1. Notation. We work in *real* Euclidean vector spaces. We let $\mathbb{S}^n$ denote the space of symmetric matrices with the trace inner product and corresponding Frobenius norm; $\mathbb{S}^n_+, \mathbb{S}^n_{++}$ are the cones of positive semidefinite, and definite, symmetric matrices of order n, respectively; denoted $S \succeq 0, S \succ 0$, respectively. We let $\mathcal{M}^n$ denote the space of square matrices of order n, also equipped with the trace inner product and Frobenius norm. We only work with real matrices in this paper. Throughout, we let $A \in \mathcal{M}^n$ be a nonsingular matrix. Hence, $M = A^T A \succ 0$.

In the literature, for $A \in \mathcal{M}^n$, the *classical condition number,* $\kappa(A) = \|A\|\|A^{-1}\| = \frac{\max \sigma_i(A)}{\min \sigma_i(A)}$, i.e., the ratio of largest to smallest singular values.

For $B \in \mathcal{M}^n$ with real eigenvalues we let

$$\lambda_i = \lambda_i(B): \quad \lambda_{\max} = \lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_{\min} = \lambda_n,$$

be the eigenvalues in nonincreasing order, and we define the functions: $\lambda_1(B) = \max_i \lambda_i(B)$; $\lambda_n(B) = \min_i \lambda_i(B)$. For our purposes, and with abuse of notation, we define the κ-condition number for *matrices with real positive eigenvalues*, B *not* necessarily symmetric, as $\kappa(B) = \frac{\lambda_{\max}(B)}{\lambda_{\min}(B)}$. We note that the product of two positive definite matrices has real positive eigenvalues, though it is not necessarily normal and not necessarily diagonalizable. We note that in this case we also have $\omega(B) = \frac{\sum_{i=1}^n \lambda_i(B)/n}{\prod_{i=1}^n \lambda_i(B)^{\frac{1}{n}}}$. We let $I_n \in \mathcal{M}^n$ be the identity matrix and let $e_n \in \mathbb{R}^n$ be the vector of ones. We often use the following matrix for a basis for the orthogonal complement $e_n^\perp$,

$$V = \frac{1}{\sqrt{2}} \begin{bmatrix} I_{n-1} \\ -e_{n-1}^T \end{bmatrix}, \quad \text{range}(V) = e_n^\perp. \tag{1.3}$$

We use $X \bullet Y$ to denote the *Hadamard product* (elementwise product) of two (compatible) matrices. For a vector $d \in \mathbb{R}^n$, $\text{Diag}(d) \in \mathbb{S}^n$ denotes the diagonal matrix formed using d. The adjoint linear transformation for $S \in \mathbb{S}^n$ is denoted $\text{Diag}^*(S) = \text{diag}(S) \in \mathbb{R}^n$. For two compatible functions f and g, we let $f \circ g$ denote the composite function.

The classical Fenchel subdifferential of a convex function h is defined as

$$\partial h(x) := \{v : \langle v, y - x \rangle \leqslant h(y) - h(x), \quad \forall y\}. \tag{1.4}$$

The Clarke subdifferential of a locally Lipschitz, not necessarily convex, function h is defined as

$$\partial_C h(x) = \text{conv}\left\{s : \exists x^k \to x, \nabla h(x^k) \text{ exists, and } \nabla h(x^k) \to s\right\}, \tag{1.5}$$

where conv denotes the convex hull of a set. It is well known that the Fenchel subdifferential and the Clarke subdifferential coincide for convex functions. Further notation is introduced below as needed.

We now recall the well-known facts about the largest and smallest eigenvalues of symmetric matrices and include a proof for completeness as it is useful further below.

LEMMA 1.1. *The maximum and minimum eigenvalue functions on* $\mathbb{S}^n$, $\lambda_{\max}, \lambda_{\min} : \mathbb{S}^n \to \mathbb{R}$, *are convex, concave, respectively.*

Proof. Letting $B \in \mathbb{S}^n$ and using the Rayleigh quotient, we get $\lambda_{\max}(B) = \max\{x^T B x : \|x\| = 1\}$, which is a maximum of linear (so convex) functions in B, and therefore is convex. We get the concavity part similarly from $\lambda_{\min}(B) = \min\{x^T B x : \|x\| = 1\}$.

Now let

$$M \in \mathbb{S}^n_{++}, d \in \mathbb{R}^n_{++}, D = \text{Diag}(d).$$

We note the following useful relations that follow by the commutativity property for eigenvalues, the fact that similarity transforms preserve the spectrum, and our definitions of κ, ω that use eigenvalues for products of positive definite matrices:

$$\lambda_i(DMD) = \lambda_i(MDD), \forall i, \quad \kappa(DMD) = \kappa(MDD), \quad \omega(DMD) = \omega(MDD). \tag{1.6}$$

2. κ-Optimal Diagonal Preconditioning. As mentioned above, preconditioning (scaling) is important in many areas of mathematics such as optimization and numerical linear algebra. (See e.g., the surveys [1,5,32]). In this section we concentrate on finding κ-optimal diagonal preconditioners D for positive definite linear

systems $Mx = b$, where $M = A^TA$ and $A \in \mathcal{M}^n$ is nonsingular. More specificially, we aim to find D that minimizes $\kappa((AD^{1/2})^TAD^{1/2}) = \kappa(D^{1/2}MD^{1/2})$.[3]

We begin in Subsection 2.1 with the eigenvalue relation in Equation (1.6). This allows us to work with the affine mapping MD in order to minimize the classical nonlinear in d formulation $\kappa(D^{1/2}MD^{1/2})$. Moreover, we provide explicit gradients and optimality conditions in Theorem 2.7. Note that [12] also considers minimizing $\kappa(D^{1/2}MD^{1/2})$, although they aim to find a diagonal preconditioner that is a convex combination of a small basis or list of diagonal preconditioners. Working with this affine mapping allows us to get expressions for derivatives for the composite functions which are used to design very efficient and scalable algorithms. In addition, our composite functions avoid the positive homogeneity in κ, ω thus improving stability.

2.1. Optimality Conditions for κ Diagonal Preconditioning. Let $M \in \mathbb{S}^n_{++}$. We denote the quadratic type scaling as follows:

$$\mathcal{D} : \mathbb{R}^n_{++} \to \mathbb{S}^n, \quad \mathcal{D}(d) = \operatorname{Diag}(d)^{1/2} M \operatorname{Diag}(d)^{1/2}.$$

With abuse of notation, we let the argument determine the function,

$$\kappa(d) = \kappa(\mathcal{D}(d)) = (\kappa \circ \mathcal{D})(d).$$

Similarly,

$$\lambda_{\max}(d) = \lambda_{\max}(\mathcal{D}(d)) = (\lambda_{\max} \circ \mathcal{D})(d), \quad \lambda_{\min}(d) = \lambda_{\min}(\mathcal{D}(d)) = (\lambda_{\min} \circ \mathcal{D})(d).$$

We use the form that is most appropriate/useful depending on the context. In the literature, e.g., [11, 12, 25], κ-optimal diagonal preconditioning refers to solving

$$d^* \in \operatorname{argmin} \left\{ \kappa(\mathcal{D}(d)) = \frac{\lambda_{\max}(\mathcal{D}(d))}{\lambda_{\min}(\mathcal{D}(d))} : d \in \mathbb{R}^n_{++} \right\}. \tag{2.1}$$

Here we restrict to $d \in \mathbb{R}^n_{++}$ as we are taking square roots.

We show below, that for our purposes, we can use the equivalent *linear in* d transformation

$$\mathring{\mathcal{D}} : \mathbb{R}^n_{++} \to \mathbb{S}^n, \ \mathring{\mathcal{D}}(d) = M \operatorname{Diag}(d). \tag{2.2}$$

With $D = \operatorname{Diag}(d)$, we first note the relationships in the eigenpairs of $\mathcal{D}(d) = D^{1/2}MD^{1/2}$ and the nonsymmetric but similar $\mathring{\mathcal{D}}(d) = MD$, and $\mathring{\mathcal{D}}(d)^T = DM$; as well we have the convexity and concavity of the maximum and minimum eigenvalues of these nonsymmetric matrices MD.[4]

LEMMA 2.1. *Let $M, D = \operatorname{Diag}(d) \in \mathbb{S}^n_{++}$, and $(u_i, \lambda_i), i = 1, \ldots, n$, be orthogonal eigenpairs of $\mathcal{D}(d)$, i.e., $\langle u_i, u_j \rangle = 0, \forall i \neq j$. Define*

$$x_i = D^{-1/2}u_i, \forall i, \ X = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix}, Y = X^{-T}, \ \Lambda = \operatorname{Diag}(\lambda).$$

Then X and Y are matrices of right and left eigenvectors of $\mathring{\mathcal{D}}(d)$, respectively, with corresponding matrix of eigenvalues Λ; and Y is given by $Y = DX(X^TDX)^{-1}$.
Moreover, $\lambda_{\max}(d) = \lambda_1(d)$ (respectively, $\lambda_{\min}(d) = \lambda_n(d)$) is a convex (respectively, concave) function of $d \in \mathbb{R}^n_{++}$.

Proof. Let $U := [u_1 \ \ldots \ u_n]$ be the corresponding matrix with *orthogonal*, not necessarily orthonormal, eigenvectors. We have $X = D^{-1/2}U$ and thus $U^TU = X^TDX$. This brings us to

$$U^{-T} = U(X^TDX)^{-1}.$$

Notice that X^TDX is a diagonal matrix with its diagonal entries given by $\|u_i\|^2$, which are strictly positive as eigenvectors are nonzero. Thus the inverse is well-defined. Now, observe that

$$\begin{aligned} Y &= X^{-T} = D^{1/2}U^{-T} \\ &= D^{1/2}U(X^TDX)^{-1} \\ &= DX(X^TDX)^{-1} \end{aligned}$$

[3] Note that we are using $\kappa(D^{1/2}MD^{1/2})$ and not the inverse $D^{-1/2}$ or D^{-1} as is customary, see e.g., (1.2). This simplifies the derivatives in the optimization problems. Moreover, the equivalent expression for $\kappa(D^{1/2}MD^{1/2})$ follows from the invariance of the spectrum after commuting matrices.

[4] These are a special case of so-called K-pd matrices in the literature, i.e., a product of two symmetric matrices where at least one is positive definite. The product of two positive definite matrices $P = AB$ arises in the optimality conditions in semidefinite programming. One of the difficulties is that the symmetrization of P is *not* necessarily positive definite, see e.g., [30].

as desired. Now,

$$\begin{aligned} D^{1/2}MD^{1/2}U = U\Lambda &\implies MDD^{-1/2}U = D^{-1/2}U\Lambda \\ &\implies MDX = X\Lambda \\ &\implies X^TDM = \Lambda X^T \\ &\implies DMX^{-T} = X^{-T}\Lambda \\ &\implies DMY = Y\Lambda. \end{aligned}$$

The second and last equation show that X and Y are right, and left, eigenvectors of MD, respectively. Moreover, we note that $M \succ 0$ has a positive definite square root and $\lambda_i(M \operatorname{Diag}(d)) = \lambda_i(M^{1/2} \operatorname{Diag}(d) M^{1/2})$. Therefore,

$$\max \lambda_i(\mathring{\mathcal{D}}(d)) = \max \lambda_i(M^{1/2} \operatorname{Diag}(d) M^{1/2}) = \max_{\|x\|=1} x^T M^{1/2} \operatorname{Diag}(d) M^{1/2} x. \qquad \square$$

As in the proof of Lemma 1.1, the latter is a maximum of linear functions in d and therefore is convex. The concavity result follows similarly.

Corollary 2.2. Let $M \in \mathbb{S}^n_{++}, d \in \mathbb{R}^n_{++}$. Then

$$\lambda_{\max}(d) = \lambda_{\max}(\mathcal{D}(d)) = \lambda_{\max}(\mathring{\mathcal{D}}(d)), \quad \lambda_{\min}(d) = \lambda_{\min}(\mathcal{D}(d)) = \lambda_{\min}(\mathring{\mathcal{D}}(d));$$

and

$$\kappa(d) = \frac{\lambda_{\max}(\mathcal{D}(d))}{\lambda_{\min}(\mathcal{D}(d))} = \frac{\lambda_{\max}(\mathring{\mathcal{D}}(d))}{\lambda_{\min}(\mathring{\mathcal{D}}(d))},$$

where recall $\mathring{\mathcal{D}}(d)$ is as in Equation (2.2).

Proof. The proof follows immediately from Lemma 2.1.

This allows us to consider the equivalent simplified view of finding a κ-optimal diagonal preconditioner, i.e., we need to solve the fractional pseudoconvex[5] minimization problem

$$\bar{d} \in \operatorname{argmin} \left\{ \kappa(d) = \frac{\lambda_{\max}(\mathring{\mathcal{D}}(d))}{\lambda_{\min}(\mathring{\mathcal{D}}(d))} : d \in \mathbb{R}^n_{++} \right\}. \tag{2.3}$$

The following Lemma 2.3 shows that for $M \succ 0$, the matrix MD having all positive eigenvalues is equivalent to the positive definiteness of D. Combining this observation with Lemma 2.1, we obtain that $\kappa(d)$ is the ratio of a convex function and a positive concave function on its domain, and is therefore pseudoconvex.

LEMMA 2.3. *Let* $M \in \mathbb{S}^n_{++}, d \in \mathbb{R}^n, D = \operatorname{Diag}(d)$. *Then*

$$\lambda_i(MD) > 0, \forall i \iff d \in \mathbb{R}^n_{++}.$$

Proof. We note that the eigenvalues of MD are the same as the eigenvalues of $M^{1/2}DM^{1/2}$. Sylvester's Lemma of inertia implies that the number of negative eigenvalues of $M^{1/2}DM^{1/2}$ is the same as that of D and so it is the same as the number of negative elements in d.

Remark 2.4. Note that the two equivalent problems Equation (2.1) and Equation (2.3) are both essentially unconstrained, and the optima are attained and characterized as stationary points in $\mathbb{R}^n_{++}$. This is not obvious but follows since we have the ratios of convex and concave functions using $\lambda_{\max}, \lambda_{\min}$ and this is over the open cone constraint $d \in \mathbb{R}^n_{++}$. If we add the constraints $d^T e_n = n, d > 0$, or equivalently that $d = e_n + Vv > 0$, with $V \in \mathbb{R}^{n \times (n-1)}$ a matrix whose columns contain a basis for $e_n^\perp$ as done above, then we have a bounded problem in $v \in \mathbb{R}^{n-1}$ and $\lambda_{\max}$ is bounded above. Bounded below away from 0 follows from applying the greedy solution to the knapsack problem with constraints $\sum_i d_i = n, d \geqslant 0$. We get

$$\lambda_{\max}(d) \quad \geqslant \quad \operatorname{tr}(MD)/n = \textstyle\sum_i M_{ii} d_i / n \geqslant \min_i M_{ii} > 0.$$

Therefore, with the denominator going to 0, we have κ going to ∞ as $d = e_n + Vv$ approaches the boundary of the simplex. That is, minimizing κ provides a *self-barrier function* for this boundary.

Moreover, $\alpha > 0 \implies \kappa(d) = \kappa(\alpha d)$, i.e., we have positive homogeneity of degree zero. Therefore, the opimal d is *not* unique. By pseudoconvexity, the optimal set is a convex set. This set can be large, see Proposition 2.8.

[5] A function $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ is said to be pseudoconvex if for all $x, y \in X$, $\nabla f(x) \cdot (y - x) \geqslant 0 \implies f(y) \geqslant f(x)$. A key property of pseudoconvex functions is that every stationary point is a global minimizer.

Lemmas 2.5 and 2.6 provide the derivative information needed for the optimality conditions.

LEMMA 2.5 ([31, (3.1)]). *Let $B : \mathbb{R} \to \mathcal{M}^n$ be differentiable with derivative $\dot{B} = \dot{B}(t)$, and, at $t = \bar{t}$, let $B(\bar{t})$ be diagonalizable with real eigenpairs (ignoring the argument $\bar{t}$)*

$$BX = X\Lambda,\ B^T Y = Y\Lambda, \quad \Lambda = \mathrm{Diag}(\lambda).$$

And make the choice $Y = X^{-T}$. Let $1 \leqslant k \leqslant n$ and λ_k be a singleton eigenvalue with right and left eigenvector x_k, y_k, respectively, taken from the corresponding columns of X, Y, respectively. Then the derivative of eigenvalues is given by

$$\dot{\lambda}_k(\bar{t}) = y_k^T(\bar{t})\dot{B}(\bar{t})x_k(\bar{t}) = \mathrm{tr}\, \dot{B} x_k y_k^T.^{6} \tag{2.4}$$

Proof. The proof is in [31, Pg 303].[7]

LEMMA 2.6. *The Fréchet derivative of the linear transformation $\mathring{\mathcal{D}}$ at d acting on Δd is (simply)*

$$\mathring{\mathcal{D}}'(d)(\Delta d) = M\,\mathrm{Diag}(\Delta d).$$

Now, let λ be a singleton eigenvalue of $\mathring{\mathcal{D}}(d)$ with a right eigenvector x. Then the gradient of the composite function at d is

$$\nabla\lambda(\mathring{\mathcal{D}}(d)) = \frac{\lambda}{x^T D x} x \bullet x.$$

Proof. The derivative of the linear transformation is clear.

Suppose as above that we have a linear transformation $\mathring{\mathcal{D}}(d)$ with singleton eigenvalue λ. Let $y = Dx/x^T Dx$ be the corresponding left eigenvector given in Lemma 2.1. Then the derivative of the composite function at d acting on Δd is

$$\begin{aligned}\langle \nabla(\lambda \circ \mathring{\mathcal{D}})(d), \Delta d\rangle &= \lambda'\left(\mathring{\mathcal{D}}'(d)(\Delta d)\right) = y^T\left(\mathring{\mathcal{D}}'(d)(\Delta d)\right)x \\ &= \tfrac{1}{x^T Dx} x^T \left(DM\,\mathrm{Diag}(\Delta d)\right)x \\ &= \tfrac{\lambda}{x^T Dx} x^T \mathrm{Diag}(\Delta d)x = \tfrac{\lambda}{x^T Dx}\langle x \bullet x, \Delta d\rangle.\end{aligned}$$

∎

Theorem 2.7 shows that at optimal preconditioning, the extreme eigenvectors must "spread mass equally" across coordinates. This balance condition is what characterizes optimal diagonal preconditioners.

THEOREM 2.7 (Characterization of κ-optimal diagonal preconditioning). *Let $M \in \mathbb{S}^n_{++}, d \in \mathbb{R}^n_{++}$ be given; let $D = \mathrm{Diag}(d)$. Then the gradient of the composite function $\kappa(d)$ is*

$$\nabla\kappa(d) = \kappa(d)\left(\frac{1}{x_1^T D x_1}(x_1 \bullet x_1) - \frac{1}{x_n^T D x_n}(x_n \bullet x_n)\right)$$

where recall that $\kappa(d) := \frac{\lambda_{\max}(\mathring{\mathcal{D}}(d))}{\lambda_{\min}(\mathring{\mathcal{D}}(d))}$. Hence, M is κ-optimally diagonally preconditioned if, and only if, the orthonormal eigenvector pair satisfies

$$x_1 \bullet x_1 = x_n \bullet x_n. \tag{2.5}$$

Equivalently, after a permutation of the elements to account for the sign,

$$x_1 = \begin{pmatrix} u \\ v \end{pmatrix},\ x_n = \begin{pmatrix} u \\ -v \end{pmatrix},\ \|u\| = \|v\|. \tag{2.6}$$

In the nonsmooth case, we can choose the normalized x_1, respectively, x_n, in the eigenspace of $\lambda_{\max}(\mathring{\mathcal{D}}(d))$, respectively $\lambda_{\min}(\mathring{\mathcal{D}}(d))$.

Proof. From Lemma 2.6, we can find the gradient as follows:

$$\begin{aligned}\nabla\kappa(d) &= \tfrac{1}{\lambda_n(d)^2}\left(\lambda_n(d)\dot{\lambda}_1(d) - \lambda_1(d)\dot{\lambda}_n(d)\right) = \tfrac{1}{\lambda_n^2}\left(\tfrac{\lambda_n\lambda_1}{x_1^T Dx_1}x_1 \bullet x_1 - \tfrac{\lambda_1\lambda_n}{x_n^T Dx_n}x_n \bullet x_n\right) \\ &= \kappa(d)\left((x_1 \bullet x_1)/(x_1^T Dx_1) - (x_n \bullet x_n)/(x_n^T Dx_n)\right).\end{aligned}$$

∎

The characterization for κ-optimally diagonally preconditioned matrix A follows from solving $\nabla\kappa(e_n) = 0$.

[6] If Y is an invertible matrix of right eigenvectors not chosen using X^{-T}, then the normalization scaling needs to be added explicitly as $\dot{\lambda}_k(\bar{t}) = y_k^T(\bar{t})\dot{B}(\bar{t})x_k(\bar{t})/(y_k(\bar{t})^T x(\bar{t}))$.

[7] The derivative of eigenvectors is included in the reference along with a useful normalization that allows for stability of the evaluations of the eigenvectors.

From (2.5) we see that at optimality, the squared magnitudes of the top and bottom eigenvectors coincide (up to permutation). This shows immediately that optimal solutions may be non-unique. This is specified using Theorem 2.7 to illustrate that the optimal set can be a large set. Proposition 2.8 shows that nonuniqueness is not pathological but structural: whenever the extreme eigenspaces do not fully determine the orthogonal complement, entire continua of optimal scalings exist. This explains why normalization is essential for numerical stability and motivates Subsection 2.3 below.

Proposition 2.8 (Nonuniqueness of κ-optimal diagonal preconditioning). Let $M \succ 0$ be a κ-optimal diagonal preconditioned matrix as given in Theorem 2.7 with $x_i, \lambda_i, i = 1, n$, being two, singleton, eigenpairs that satisfy the optimality conditions in Theorem 2.7. Thus we have

$$\lambda_1 > \lambda_2 \geqslant \ldots \geqslant \lambda_{n-1} > \lambda_n > 0, \quad \lambda = (\lambda_i) \in \mathbb{R}^n_{++}, \, \Lambda = \operatorname{Diag}(\lambda),$$

and we let $Q = \begin{bmatrix} x_1 & \bar{Q} & x_n \end{bmatrix}$ be an orthogonal matrix and $M = Q\Lambda Q^T$ from the spectral theorem. Let V be the basis for $e_n^{\perp}$ defined in Equation (1.3). Suppose in addition that the *lack* of *strict complementarity condition* holds, i.e., there exists v such that

$$0 \neq v \in \{u \in \mathbb{R}^{n-1} : 0 = Vu \bullet x_1 = Vu \bullet x_n\}. \tag{2.7}$$

Then with $\Delta D = \operatorname{Diag}(Vv)$, there exists $\epsilon > 0$ such that

$$\kappa(M) = \kappa\left((I + t\Delta D)M(I + t\Delta D)\right), \, \forall |t| \leqslant \epsilon.$$

Proof. The result follows from expanding

$$(I + \epsilon\Delta D)M(I + \epsilon\Delta D) = M + O(\epsilon)$$

and using the continuity of eigenvalues and the fact that the optimality conditions imply

$$\begin{array}{rcl} Vv \bullet x_i = 0, i = 1, n & \iff & \operatorname{Diag}(Vv)x_i = 0, i = 1, n \\ & \iff & \Delta D x_i = 0, i = 1, n. \end{array}$$

Specifically, let

$$M_2 = \bar{Q} \operatorname{Diag}((\lambda_2, \ldots, \lambda_{n-1})^T)\bar{Q}^T, \quad M_1 = M - M_2.$$

The above equation then implies $\Delta D M_1 = M_1 \Delta D = 0$. Moreover, let $D = (I + \epsilon\Delta D)$, then

$$DMD = M_1 + DM_2D = M + \epsilon\Delta D M_2 + \epsilon M_2 \Delta D + \epsilon^2 \Delta D M_2 \Delta D. \qquad \square$$

Since $\operatorname{range}(\Delta D) \subseteq \operatorname{null}(M_1) = \operatorname{range}(M_2)$, the range of the perturbation of M above is restricted to the eigenspace of M_2, i.e., to the $\operatorname{span}(\{x_2, x_3, \ldots, x_{n-1}\})$. This means that after the perturbation, with $\epsilon > 0$ sufficiently small, λ_1 and λ_n remain the largest and smallest eigenvalues of DMD respectively. As stated above, we are using the continuity of eigenvalues and the orthogonality $M_1 M_2 = 0$ that arises using the spectral theorem and $M_1, M_2 \succeq 0$.

Note that if condition (2.7) holds, then we get a nonsingleton set in $\mathbb{R}^{n-1}$ of solutions. Moreover, the structure of V implies that (2.7) restricts the support of the eigenspace $\operatorname{span}(\{x_1, x_n\})$.[8]

2.2. A Projected Subgradient Method for Minimizing $\kappa(d)$. In Theorem 2.7, we derived the gradient and optimality conditions for minimizing the pseudoconvex function $\kappa(d)$ in Equation (2.3), over the open set $d > 0$. Convergence of subgradient methods for pseudoconvex minimization typically requires optimizing over a closed set. Hence, we consider the following optimization problem:

$$\bar{d} \in \operatorname{argmin} \left\{ \kappa(d) = \frac{\lambda_{\max}(\mathring{\mathcal{D}}(d))}{\lambda_{\min}(\mathring{\mathcal{D}}(d))} : d \in \Omega := \{d : d \geqslant \delta e_n\} \right\}, \tag{2.8}$$

where in this subsection $\kappa(d) := \infty$ for $d \notin \Omega$ and $\delta \in (0, 1)$ is small.

We now exploit the pseudoconvex structure that guarantees all stationary points are global minimizers making subgradient methods natural. We first treat the unconstrained formulation in Algorithm 2.1 and address homogeneity in Subsection 2.3.

[8] Necessity is still an open problem.

Algorithm 2.1 A Subgradient Method for Minimizing $\kappa(d)$ over Ω

Inputs: symmetric positive definite matrix $M \succ 0$; sequence of positive stepsizes $\{t_k\} \to 0$ with $\sum_{k=1}^{\infty} t_k = \infty$, scalar $\delta \in (0,1)$; tolerance tol; rule for the stopping criterion, stopcrit.

1: set $k \leftarrow 0$;
2: set stopcrit $\leftarrow \infty$;
3: set $d_1 = e_n \in \mathbb{R}^n$;
4: **while** stopcrit>tol **do**
5: set $k \leftarrow k+1$;
6: compute min eigenpair (λ_n^k, x_n^k) and max eigenpair (λ_1^k, x_1^k) of $M\,\mathrm{Diag}(d_k)$;
7: compute direction

$$s_k = \frac{\lambda_1^k}{\lambda_n^k}\left(\frac{1}{\langle x_1^k, d_k \bullet x_1^k\rangle}\left(x_1^k \bullet x_1^k\right) - \frac{1}{\langle x_n^k, d_k \bullet x_n^k\rangle}\left(x_n^k \bullet x_n^k\right)\right); \tag{2.9}$$

8: perform projected gradient step

$$d_{k+1} = \max\left\{d_k - t_k \frac{s_k}{\|s_k\|}, \delta e_n\right\}; \tag{2.10}$$

9: update stopcrit;
10: **end while**(main outer loop)

Output: $\hat{D} := \mathrm{Diag}(d_{k+1})$.

Remark 2.9. For Algorithm 2.1, we use a popular choice for stepsize sequence $\{t_k\}$ in subgradient methods, $t_k = 1/k$, where k is the iteration index [19]. We note that since Algorithm 2.1 is a subgradient method, in general it is not a descent method. Finally, the algorithm is general in that it does not specify the stopping rule stopcrit. There are several possible rules that the user can employ in practice for updating stopcrit in step 9. For example, at every iteration the user can take stopcrit as $\|s_k\|$ or $|\kappa(d_{k+1}) - \kappa(d_k)|$.

We now briefly describe each of the steps of Algorithm 2.1. First, step 6 computes a maximal and minimal eigenpair of $M\,\mathrm{Diag}(d_k)$ to construct the search direction s_k. It should be noted that the user never has to form the matrix $M\,\mathrm{Diag}(d_k)$ to compute the eigenpairs. Instead, the user only has to construct subroutines which compute $M\,\mathrm{Diag}(d_k) * x$ and $(M\,\mathrm{Diag}(d_k))\backslash x$ efficiently, where $x \in \mathbb{R}^n$. Second, it will be shown in Subsection 2.2.1 that the direction s_k computed in step 7 lies in the quasisubdifferential (also to be defined in Subsection 2.2.1) of $\kappa(d_k)$. Finally, using this direction, step 8 performs the projected gradient update $d_{k+1} = \Pi_\Omega(d_k - t_k \frac{s_k}{\|s_k\|})$, where $\Pi_\Omega(x) := \mathrm{argmin}_\Omega \|x - \cdot\|$ denotes the projection onto Ω.

2.2.1. Asymptotic Convergence Analysis. Our convergence analysis of Algorithm 2.1 relies mostly on [19] where efficient subgradient methods for minimizing quasiconvex functions are presented. Under various assumptions in addition to quasiconvexity, the author establishes asymptotic convergence of subgradient methods with decaying stepsizes. Since our function $\kappa(d)$ is pseudoconvex, it is quasiconvex. For a more detailed discussion related to the assumptions of Kiwiel [19] and how our set-up satisfies these assumptions, the reader is referred to Appendix A. In the remaining part of this subsection, we show that Algorithm 2.1 asymptotically converges by showing that it is an instance of the subgradient framework proposed by Kiwiel.

First, we need to define a special subdifferential called the quasisubdifferential for a quasiconvex function f. Define the strict sublevel set or inner slice of f as:

$$\bar{S}(x) := \left\{y \in \mathrm{int}\,\bar{D} : f(y) < f(x)\right\}, \tag{2.11}$$

where $\mathrm{int}\,\bar{D}$ denotes the interior of the domain of f. The quasisubdifferential of a quasiconvex function f relative to the above sublevel set is defined as

$$\bar{\partial}^\circ f(x) := \left\{g : \langle g, y - x\rangle < 0, \quad \forall y \in \bar{S}(x)\right\}. \tag{2.12}$$

To minimize a quasiconvex function f over a closed convex set X, Kiwiel proposes in [19] the following basic subgradient algorithm:

$$x_{k+1} := \Pi_X(x_k - t_k \hat{g}_k), \quad \hat{g}_k := g_k/\|g_k\|, \quad g_k \in \bar{\partial}^\circ f(x_k), \quad k = 1, 2, \ldots, \quad x_1 \in X,$$

where $t_k > 0$ are the stepsizes.

Hence, we need to characterize vectors that lie in the quasisubdifferential of $\kappa(d)$ to show that Algorithm 2.1 is an instance of Kiwiel's subgradient framework. The result below presents one characterization of vectors that lie in the quasisubdifferential of fractional programs like the one in our set-up Equation (2.8).

LEMMA 2.10. *Suppose $f(x) := a(x)/b(x)$ for all $x \in X$ and $f(x) := \infty$ for $x \notin X$, where $a(x)$ is a convex function that is positive on X, $b(x)$ is a concave function that is positive on X. Let $\bar{D}^b$ denote the domain of b. If for $x \in X \cap \bar{D}^b$, $B := 1/b(x)$ and $A := a(x)/b^2(x)$, then*

$$B[\partial a(x)] + A[\partial(-b)(x)] \subseteq \bar{\partial}^\circ f(x).$$

Proof. Let $x \in X \cap \bar{D}^b$ and consider B and A as in the assumptions of the lemma. It follows from the fact that a and $-b$ are convex functions, B and A are positive scalars, and Fenchel subdifferential calculus rules that

$$B[\partial a(x)] + A[\partial(-b)(x)] = \partial(Ba)(x) + \partial(-Ab)(x) \subseteq \partial(Ba - Ab)(x).$$

Now, let $g \in B[\partial a(x)] + A[\partial(-b)(x)] \subseteq \partial(Ba - Ab)(x)$, and suppose $y \in \bar{S}(x)$, i.e., y is in the interior of the domain of f and $f(y) < f(x)$. It then follows that

$$\begin{aligned}\langle g, y - x\rangle &\leqslant Ba(y) - Ab(y) - Ba(x) + Ab(x)\\ &= \frac{a(y)}{b(x)} - \frac{a(x)b(y)}{b^2(x)} < 0,\end{aligned}$$

where the first inequality follows from the definition of Fenchel subdifferential, the equality follows from the definitions of A and B, and the last inequality follows from the fact that $a(y)/b(y) < a(x)/b(x)$ and $b(y)$ and $b(x)$ are positive. It then follows from the definition of quasisubdifferential in Equation (2.12) that $g \in \bar{\partial}^\circ f(x)$. ∎

Corollary 2.11 constructs a vector that lies in the quasisubdifferential of $\kappa(d)$.

COROLLARY 2.11. *Consider $M \succ 0$ as in Equation (2.8) and $d \in \Omega \cap \operatorname{int} \bar{D}^{\lambda_{\min}}$, where $\bar{D}^{\lambda_{\min}}$ is the domain of $\lambda_{\min}(\mathring{\mathcal{D}}(d))$. Let (λ_1, x_1) and (λ_n, x_n) be a maximal and minimal eigenpair of $\mathring{\mathcal{D}}(d)$, respectively. Then, it holds that*

$$\frac{\lambda_1}{\lambda_n}\left(\frac{1}{x_1^T(d \bullet x_1)}(x_1 \bullet x_1) - \frac{1}{x_n^T(d \bullet x_n)}(x_n \bullet x_n)\right) \in \bar{\partial}^\circ\left(\kappa(d)\right).$$

Proof. It follows from Lemma 2.1 that $\lambda_{\max}(\mathring{\mathcal{D}}(d))$ and $\lambda_{\min}(\mathring{\mathcal{D}}(d))$ are convex and concave functions of d, respectively. Also, it is easy to see that $\lambda_{\max}(\mathring{\mathcal{D}}(d))$ and $\lambda_{\min}(\mathring{\mathcal{D}}(d))$ are positive on Ω. Hence, it follows from Lemma 2.10 that

$$\frac{1}{\lambda_{\min}(\mathring{\mathcal{D}}(d))}\partial\lambda_{\max}(\mathring{\mathcal{D}}(d)) + \frac{\lambda_{\max}(\mathring{\mathcal{D}}(d))}{\lambda^2_{\min}(\mathring{\mathcal{D}}(d))}\partial\left(-\lambda_{\min}(\mathring{\mathcal{D}}(d))\right) \subseteq \bar{\partial}^\circ\left(\kappa(d)\right) \tag{2.13}$$

holds for $d \in \Omega \cap \operatorname{int} \bar{D}^{\lambda_{\min}}$. Moreover, it follows from Lemma 2.6, the definition of Clarke subdifferential in Equation (1.5), and the fact that the Fenchel and Clarke subdifferentials coincide for convex functions that the following inclusions hold.

$$\frac{\lambda_1}{x_1^T\,(d \bullet x_1)}\,(x_1 \bullet x_1) \in \partial\lambda_{\max}(\mathring{\mathcal{D}}(d)), \quad -\frac{\lambda_n}{x_n^T\,(d \bullet x_n)}\,(x_n \bullet x_n) \in \partial\left(-\lambda_{\min}(\mathring{\mathcal{D}}(d))\right), \tag{2.14}$$

where here (λ_1, x_1) and (λ_n, x_n) are maximal and minimal eigenpairs of $\mathring{\mathcal{D}}(d)$, respectively. The result then follows from Equation (2.13) and Equation (2.14). ∎

Remark 2.12. It follows from Corollary 2.11 that the update rule Equation (2.10) in step 8 is of the form

$$d_{k+1} = \Pi_\Omega\left(d_k - t_k\frac{s_k}{\|s_k\|}\right), \text{ where } s_k \in \bar{\partial}^\circ\left(\kappa(d_k)\right).$$

We are now ready to present the main theorem that shows Algorithm 2.1 converges asymptotically.

THEOREM 2.13 (Asymptotic convergence). *Let $\{d_k\}$ be generated by Algorithm 2.1. Let*

$$\kappa_* := \min\{\kappa(d) : d \in \Omega\}; \quad \kappa_*^k = \min\{\kappa(d_j), j = 1\ldots, k\}.$$

Then

$$\underline{\lim}_{k\to\infty}\kappa(d_k) = \kappa_*; \quad \text{and } \kappa_*^k \downarrow \kappa_*.$$

Proof. It follows from the last remark in Lemma 2.1 and the definitions of $\kappa(d)$ and Ω in Equation (2.8), that $\lambda_{\max}(\mathring{\mathcal{D}}(d))$ (resp. $\lambda_{\min}(\mathring{\mathcal{D}}(d))$) is a convex (resp. concave) function that is positive on Ω. It then follows from this observation, Lemma A.2, and the definition of $\kappa(d)$, that assumptions **A1-A4** in Appendix A hold for the minimization problem in Equation (2.8). Clearly, also assumption **A5** in Appendix A also holds since Ω in Equation (2.8) is a closed set and the intersection of Ω with the interior of the domain of $\kappa(d)$ is clearly nonempty. The result of the theorem then immediately follows from this observation, Remark 2.12, the facts that $t_k \to 0$, and $\sum_{k=1}^{\infty} t_k = \infty$, and Theorem 1 in [19].

2.3. Avoiding Positive Homogeneity to Ensure Well-Posedness in Minimizing $\kappa(d)$. As mentioned in Remark 2.4, $\kappa(d)$ is a positively homogenous function (of degree zero). Thus there are multiple optimal solutions and our problem is *Hadamard ill-posed*. From a computational stability perspective, it is more efficient to minimize an equivalent smaller dimensional formulation that is not positively homogeneous. With this in mind, we let $M \succ 0$ and $V \in \mathbb{R}^{n\times(n-1)}$ be a matrix whose columns form a basis for $e_n^{\perp}$, where $e_n \in \mathbb{R}^n$ is the vector of all ones. We consider the function

$$\mathring{\mathcal{V}}(v) := M\,\mathrm{Diag}(e_n + Vv), \tag{2.15}$$

where $v \in \mathbb{R}^{n-1}$. In this subsection, we consider the following formulation

$$\min\left\{\kappa(v) := \frac{\lambda_{\max}(\mathring{\mathcal{V}}(v))}{\lambda_{\min}(\mathring{\mathcal{V}}(v))} : e_n + Vv \geqslant \hat{\delta} e_n,\ v \in \mathbb{R}^{n-1}\right\}, \tag{2.16}$$

where $\kappa(v) := \infty$ if $e_n + Vv < \hat{\delta} e_n$ and $\hat{\delta} \in (0,1)$ is a scalar. It is easy to see that with the choice of V in (1.3), the above Equation (2.16) can be rewritten as

$$\min\left\{\kappa(v) : v \in \hat{\Omega}\right\}, \tag{2.17}$$

where

$$\hat{\Omega} := \left\{v \in \mathbb{R}^{n-1} : \sum_{i=1}^{n-1} v_i \leqslant -\sqrt{2}(\hat{\delta}-1), \quad v_i \geqslant \sqrt{2}(\hat{\delta}-1), \quad i = 1,\dots n-1\right\}. \tag{2.18}$$

Note that the set $\hat{\Omega}$ is convex and compact. Projecting onto it is also easy since it is just a simplex. Also, since $\hat{\Omega}$ is bounded, we will be able to get nonasymptotic convergence guarantees for the subgradient method that we propose to minimize Equation (2.17).

The following lemma will be useful for developing our subgradient method. It gives useful characterizations of the derivatives of $\mathring{\mathcal{V}}(v)$ and $\kappa(v)$ in the smooth setting when the maximum and minimum eigenvalues of $\mathring{\mathcal{V}}(v)$ have multiplicity one.

LEMMA 2.14 (Derivatives of $\mathring{\mathcal{V}}(v), \kappa(v)$). *Let $M \succ 0, v \in \mathbb{R}^{n-1}$ be given and set $w = e_n + Vv \in \mathbb{R}^n_{++}$ and $D = \mathrm{Diag}(w)$. Also, let (λ_1, x_1) and (λ_n, x_n) be maximal and minimal eigenpairs of $\mathring{\mathcal{V}}(v)$, respectively, where λ_1 and λ_n have multiplicity one and x_1 and x_n are assumed to be normalized. Then the following hold:*

1 The derivative at v acting on $\Delta v \in \mathbb{R}^{n-1}$ is $\dot{\mathring{\mathcal{V}}}(v)(\Delta v) := \mathring{\mathcal{V}}'(v)(\Delta v) = M\,\mathrm{Diag}(V\Delta v)$.

2 The gradient of the composite function $\kappa(v) := \kappa(\mathring{\mathcal{V}}(v))$ is

$$\nabla\kappa(v) = \kappa(v) V^T \left(\frac{1}{x_1^T(w \bullet x_1)}(x_1 \bullet x_1) - \frac{1}{x_n^T(w \bullet x_n)}(x_n \bullet x_n)\right). \tag{2.19}$$

Proof. 1 The proof follows from the function being affine.

2 For a singleton eigenvalue λ with normalized right eigenvector x, we have the left eigenvector $y = Dx/(x^T Dx)$ as in Lemma 2.1, which satisfies $x^T y = 1$. Then

$$\begin{aligned}
\langle \nabla(\lambda \circ \mathring{\mathcal{V}})(v), \Delta v\rangle &= y^T \dot{\mathring{\mathcal{V}}}(v)(\Delta v)x = y^T(M\,\mathrm{Diag}(V\Delta v))x \\
&= \langle V^T \mathrm{diag}(\lambda D^{-1} y x^T), \Delta v\rangle \\
&= \tfrac{1}{x^T Dx}\langle V^T \mathrm{diag}(\lambda D^{-1} D x x^T), \Delta v\rangle \\
&= \tfrac{1}{x^T(w\bullet x)}\langle V^T \mathrm{diag}(\lambda x x^T), \Delta v\rangle = \tfrac{1}{x^T(w\bullet x)}\langle \lambda V^T(x \bullet x), \Delta v\rangle.
\end{aligned}$$

Therefore the gradient of $\kappa(v) := \kappa(\mathring{\mathcal{V}}(v))$ is:

$$\begin{aligned} \nabla\kappa(v) &= \frac{\lambda_n\dot{\lambda}_1 - \lambda_1\dot{\lambda}_n}{\lambda_n^2} = \frac{1}{\lambda_n^2}\left(\frac{\lambda_n}{x_1^T(w\bullet x_1)}\lambda_1 V^T(x_1\bullet x_1) - \frac{\lambda_1}{x_n^T(w\bullet x_n)}\lambda_n V^T(x_n\bullet x_n)\right) \\ &= \kappa(v)V^T\left(\frac{1}{x_1^T(w\bullet x_1)}(x_1\bullet x_1) - \frac{1}{x_n^T(w\bullet x_n)}(x_n\bullet x_n)\right). \end{aligned} \tag{2.20}$$

∎

The following remark shows that, as in Lemma 2.14, $\nabla\kappa(v)$ lies in the quasisubdifferential of $\kappa(v)$.

Remark 2.15. Let $M \succ 0$ and $v \in \mathbb{R}^{n-1}$. Also, suppose that $w = e_n + Vv$ and let (x_i, λ_i), $i = 1, n$, be eigenpairs of $\mathring{\mathcal{V}}(v)$, where $\|x_i\| = 1$. It then follows from Lemma 2.10 and a similar argument as in the proof of Corollary 2.11 that

$$\kappa(v)V^T\left(\frac{1}{x_1^T(w\bullet x_1)}(x_1\bullet x_1) - \frac{1}{x_n^T(w\bullet x_n)}(x_n\bullet x_n)\right) \in \bar{\partial}^\circ\left(\kappa(v)\right) \tag{2.21}$$

where recall that $\kappa(v) := \kappa(\mathring{\mathcal{V}}(v))$.

We now present our subgradient algorithm for minimizing Equation (2.17).

Algorithm 2.2 A Subgradient Method for Minimizing $\kappa(v) := \kappa(\mathring{\mathcal{V}}(v))$ over $\hat{\Omega}$

Inputs: symmetric positive definite matrix $M \succ 0$; $V \in \mathbb{R}^{n\times(n-1)}$ a basis matrix for the orthogonal complement $e^\perp$; sequence of stepsizes $\{t_k\} = 1/\sqrt{k}$; scalar $\hat{\delta} \in (0,1)$; a tolerance tol> 0; a rule for the stopping criterion, stopcrit.

1: set $k \leftarrow 0$;
2: set stopcrit $\leftarrow \infty$;
3: set $v_1 = 0 \in \mathbb{R}^{n-1}$ and $w_1 = e_n \in \mathbb{R}^n_+$;
4: **while** stopcrit>tol **do**
5: set $k \leftarrow k + 1$.
6: compute min eigenpair (λ_n^k, x_n^k) and max eigenpair (λ_1^k, x_1^k) of $M\,\mathrm{Diag}(w_k)$;
7: compute direction

$$g_k = \frac{\lambda_1^k}{\lambda_n^k}V^T\left(\frac{1}{\langle x_1^k, w_k\bullet x_1^k\rangle}\left(x_1^k\bullet x_1^k\right) - \frac{1}{\langle x_n^k, w_k\bullet x_n^k\rangle}\left(x_n^k\bullet x_n^k\right)\right) \tag{2.22}$$

8: perform projected gradient step

$$v_{k+1} = \Pi_{\hat{\Omega}}\left(v_k - t_k\frac{g_k}{\|g_k\|}\right) \tag{2.23}$$

where $\hat{\Omega}$ is as in Equation (2.18) and set

$$w_{k+1} = e + Vv_{k+1}; \tag{2.24}$$

9: update stopcrit;
10: **end while**(main outer loop)

Output: $\hat{D} := \mathrm{Diag}(w_{k+1})$.

Several remarks about Algorithm 2.2 are now given. First, the matrix V does not need to be stored in memory and is actually not needed as input. All the user needs to input is a subroutine that outputs Vv given a vector $v \in \mathbb{R}^{n-1}$. Likewise, the matrix $M\,\mathrm{Diag}(w_k)$ does not need to be stored. Second, it follows from Remark 2.15 that $g_k \in \bar{\partial}^\circ\left(\kappa(v_k)\right)$, which is defined in Equation (2.12). Finally, the projected gradient step in Equation (2.23) can be performed very efficiently since projecting onto $\hat{\Omega}$ just involves computing the root of a simple equation.

2.3.1. Nonasymptotic Convergence Rate Analysis for Minimizing $\kappa(v)$ on $\hat{\Omega}$. In this subsection, we show a nonasymptotic convergence rate for Algorithm 2.2 for minimizing $\kappa(v)$ over $\hat{\Omega}$. More specifically, we show that

$$\min_{1\leqslant k\leqslant K}\kappa(v_k) - \kappa_* \leqslant \epsilon$$

holds for $K = \mathcal{O}(1/\epsilon^2)$, where $\kappa_* = \min_{v\in\hat{\Omega}}\kappa(v)$. Our nonasymptotic convergence analysis of Algorithm 2.2 relies mostly on the results from [19] and [15]. The function $\kappa(v)$ is pseudoconvex since it is the ratio of a

convex function and a positive concave function. Also, observe that the set $\hat{\Omega}$ is just a box so it is a convex compact set that is easy to project onto. The only other assumption that needs to be verified to be able to show a nonasymptotic convergence rate of Algorithm 2.2 is that the function $\kappa(v)$ is Lipschitz continuous on $\hat{\Omega}$. This result is proved in the following proposition.

Proposition 2.16. The function $\kappa(v)$ is Lipschitz continuous on $\hat{\Omega}$, i.e.,

$$\|\kappa(v_1) - \kappa(v_2)\| \leqslant L\|v_1 - v_2\|, \quad \forall v_1, v_2 \in \hat{\Omega}$$

where $L > 0$ and $\hat{\Omega}$ is as in Equation (2.18).

Proof. First, it is easy to see that $\lambda_{\max}(\mathring{\mathcal{V}}(v))$ (resp. $\lambda_{\min}(\mathring{\mathcal{V}}(v))$) is a convex (resp. concave) function. It then follows from this observation, the fact $\hat{\Omega}$ is a compact set that is contained in the domain of both functions, and Proposition A.48(b) in [23] that $\lambda_{\max}(\mathring{\mathcal{V}}(v))$ (resp. $\lambda_{\min}(\mathring{\mathcal{V}}(v))$) is L_1-Lipschitz (resp. L_2-Lipschitz) on $\hat{\Omega}$. Consider now the composite function $h(v) = g(\lambda_{\min}(\mathring{\mathcal{V}}(v)))$ where $g(y) = 1/y$. The above conclusion and the fact that $g(y) = 1/y$ is Lipchitz continuous on any positive open interval then imply that h is L_3-Lipschitz continuous on $\hat{\Omega}$, where $L_3 > 0$.

We now show that $\kappa(v)$ is Lipschitz continuous. First, it holds that $|\lambda_{\max}(v)| \leqslant M_1$ and $|h(v)| \leqslant M_2$ for $v \in \hat{\Omega}$ since a Lipschitz function on a compact set is bounded. Then, for any $v_1 \in \hat{\Omega}$ and $v_2 \in \hat{\Omega}$, the following holds:

$$\begin{aligned}
\|\kappa(\mathring{\mathcal{V}}(v_1)) - \kappa(\mathring{\mathcal{V}}(v_2))\| &= \|\lambda_{\max}(\mathring{\mathcal{V}}(v_1))h(v_1) - \lambda_{\max}(\mathring{\mathcal{V}}(v_2))h(v_2)\| \\
&= \|\lambda_{\max}(\mathring{\mathcal{V}}(v_1))h(v_1) - \lambda_{\max}(\mathring{\mathcal{V}}(v_1))h(v_2) + \lambda_{\max}(\mathring{\mathcal{V}}(v_1))h(v_2) - \lambda_{\max}(v_2)h(v_2)\| \\
&\leqslant M_1\|h(v_1) - h(v_2)\| + M_2\|\lambda_{\max}(\mathring{\mathcal{V}}(v_1)) - \lambda_{\max}(\mathring{\mathcal{V}}(v_2))\| \\
&\leqslant M_1L_3\|v_1 - v_2\| + M_2L_1\|v_1 - v_2\| = (M_1L_3 + M_2L_1)\|v_1 - v_2\|,
\end{aligned}$$

∎

which immediately implies the statement of the proposition with $L = M_1L_3 + M_2L_1$.

We now state Theorem 2.17, which displays that Algorithm 2.2 is able to find an ϵ-approximate optimal solution of Equation (2.17) with a sublinear $\mathcal{O}(1/\epsilon^2)$ rate of convergence.

THEOREM 2.17. *Let $\epsilon > 0$ be a given tolerance and suppose that $K = \mathcal{O}(1/\epsilon^2)$. It then holds that*

$$\min_{1 \leqslant k \leqslant K} \kappa(v_k) - \kappa_* \leqslant \epsilon, \tag{2.25}$$

where $\kappa_* = \min_{v \in \hat{\Omega}} \kappa(v)$ and $\kappa(v) := \kappa(\mathring{\mathcal{V}}(v))$.

Proof. It is immediate to see that $\hat{\Omega}$ is a convex compact set and that $\kappa(v)$ is a quasiconvex function since it is the ratio of a convex and a positive concave function. It follows from Proposition 2.16 that $\kappa(v)$ is Lipschitz continuous on $\hat{\Omega}$. Finally, it follows from Remark 2.15 that the update Equation (2.23) in step 8 of Algorithm 2.2 is of the form $v_{k+1} = \Pi_{\hat{\Omega}}\left(v_k - t_k \frac{g_k}{\|g_k\|}\right)$ where $g_k \in \bar{\partial}^\circ(\kappa(v_k))$. Hence, it follows from these observations, the fact that $\{t_k\} = 1/\sqrt{k}$, and Theorem 3.2(ii) in [15] with $s = 1/2$, $\delta = \epsilon$, and $p = 1$ that the statement of Theorem 2.17 holds.

3. ω-Optimal Structured Preconditioning. This section focuses on obtaining ω-optimal preconditioners of special structure for finding least squares solutions of $Ax = b$, where A might not be symmetric nor square. In what follows, A will be assumed to have full column rank, so that the matrix $M := A^TA$ is positive definite. Recall that for $M > 0$, we define $\omega(M)$ as

$$\omega(M) = \frac{\mathrm{tr}(M)/n}{\det(M)^{1/n}},$$

i.e., it is the ratio of the arithmetic to geometric means of the eigenvalues of M. Below, this measure will be used to study the conditioning of systems $Ax = b$, when A is not necessarily positive definite.

This section is divided into three main subsections. First, Subsection 3.1.1 focuses on right-sided ω-optimal diagonal preconditioning, for both symmetric positive definite and full rank matrices. We treat left-sided ω-optimal diagonal preconditioning for invertible matrices in Subsection 3.1.2. Then in Subsection 3.2 we

TABLE 3.1
Relation of ω-Optimal Preconditioners to the Literature

Special Structure	Preconditioner	Location	Literature
Right Diagonal	Column Normalization	Proposition 3.1	[7, 8]
Diagonal for Symmetric $M \succ 0$	Jacobi Preconditioner	Corollary 3.2	[16]
Left Diagonal	Row Normalization	Theorem 3.3	
Two-Sided Diagonal	Sinkhorn–Knopp/Matrix Balancing	Theorem 3.6	[20, 28, 29]
Right Block-Diagonal	Block QR	Corollary 3.11	[8, 17]
Left Block-Diagonal	Characterization	Theorem 3.12	

focus on two-sided ω-optimal diagonal preconditioning. Finally, Subsection 3.3 considers ω-optimal block diagonal preconditioning for non-square matrices. In particular, we focus on preconditioners with triangular blocks.

Interestingly, we see that many popular preconditioners used in the literature can be found as being ω-optimal with special structure constraints, as illustrated in Table 3.1.

3.1. Right- and Left-Sided Diagonal. Suppose that $\operatorname{diag}(A) \neq 0$. The classical Jacobi preconditioner [16], [27, Sect. 10.2] uses:

$$P^{-1}A = P^{-1}b, \quad P = \operatorname{Diag}(\operatorname{diag}(A)). \tag{3.1}$$

We note that this can be found by using: a splitting $A = M - N$ so that $A = \operatorname{Diag}(\operatorname{diag}(A)) - N$; or the following variational problem that implicitly finds the best approximation of the identity using diagonal matrices:

$$\min_d \|A - \operatorname{Diag}(d)\|_F, \quad \operatorname{Diag}(d)^{-1}A \approx I.$$

We now consider variational problems using ω.

3.1.1. Right-Sided Diagonal. The following result shows that a ω-optimal right-sided diagonal scaling of an overdetermined full rank matrix A is formed using $(\pm)$ the reciprocal of the column norms of A, see [7].

Proposition 3.1. Let A be $m \times n$ full column rank, and $D = \operatorname{Diag}(\operatorname{diag}(A^TA))^{-1}$. Then a ω-optimal diagonal right scaling is $AD^{1/2}$, i.e.,

$$D \in \operatorname{argmin} \left\{ \omega(D^{1/2}A^TAD^{1/2}) : \operatorname{diag}(D) \in \mathbb{R}^n_{++} \right\}.$$

Proof. The proof of the result can be found in [7, Proposition 2.1(v)] and a modified proof in [17, Proposition 2.1(3)].

Corollary 3.2. Let $M \in \mathbb{S}^n_{++}$. Then the Jacobi preconditioner, $D = \operatorname{Diag}(\operatorname{diag}(M))^{-1}$, is the ω-optimal diagonal scaling, i.e.,

$$D \in \operatorname{argmin} \left\{ \omega(D^{1/2}MD^{1/2}) : \operatorname{diag}(D) \in \mathbb{R}^n_{++} \right\}.$$

Proof. The proof is immediate from Proposition 3.1 with A^TA replaced by $M \in \mathbb{S}^n_{++}$.

3.1.2. Left-Sided Diagonal. In this section we let $A \in \mathcal{M}^n$ be invertible, and we consider the optimal left-sided diagonal preconditioning problem:

$$\begin{array}{ll} \min & \omega\big((\operatorname{Diag}(c)^{1/2}A)^T \operatorname{Diag}(c)^{1/2}A\big) \\ \text{s.t.} & c \in \mathbb{R}^n_{++}. \end{array} \tag{3.2}$$

For simplicity, let $C := \operatorname{Diag}(c)$. We can characterize the ω-optimal left-sided diagonal preconditioner using the above results in Subsection 3.1.1 for the right preconditioner.

THEOREM 3.3. *Let $A \in \mathcal{M}^n$ be nonsingular. Then an ω-optimal left-sided diagonal preconditioner minimizing (3.2) is given by $C = \operatorname{Diag}(\operatorname{diag}(AA^T))^{-1}$.*

Proof. Note that

$$\omega\big((\operatorname{Diag}(c)^{1/2}A)^T \operatorname{Diag}(c)^{1/2}A\big) = \omega\big(A^T \operatorname{Diag}(c)A\big) = \omega\big(\operatorname{Diag}(c)^{1/2}AA^T \operatorname{Diag}(c)^{1/2}\big).$$

Hence, Proposition 3.1 and the above equivalence implies that

$$\begin{aligned} C = \operatorname{Diag}(\operatorname{diag}(AA^T))^{-1} &\iff C \in \operatorname{argmin} \left\{ \omega(C^{1/2}AA^TC^{1/2}) : C = \operatorname{diag}(c) \in \mathbb{S}^n_{++} \right\} \\ &\iff C \in \operatorname{argmin} \left\{ \omega\big((\operatorname{Diag}(c)^{1/2}A)^T \operatorname{Diag}(c)^{1/2}A\big) : C = \operatorname{diag}(c) \in \mathbb{S}^n_{++} \right\}. \end{aligned}$$

∎

3.2. Two-Sided Diagonal. We now consider the problem of finding the two-sided ω-optimal diagonal scaling for nonsingular matrices $A \in \mathcal{M}^n$. This subsection is broken up into two smaller parts. The first part derives the optimality conditions for the two-sided ω-optimal scaling problem. The second part presents an iterative matrix balancing scheme that reduces ω at every iteration and whose output is proved to be a stationary point of the two-sided ω-optimal scaling problem.

3.2.1. Optimality Conditions for Two-Sided Problem. This part presents the formulation of the two-sided ω-optimal diagonal scaling problem and derives its optimality conditions. Namely, we consider

$$\begin{array}{ll} \min & \omega((\mathrm{Diag}(c)^{1/2} A \,\mathrm{Diag}(d)^{1/2})^T (\mathrm{Diag}(c)^{1/2} A \,\mathrm{Diag}(d)^{1/2})) \\ \text{s.t.} & c, d \in \mathbb{R}^n_{++}, \end{array} \tag{3.3}$$

where $A \in \mathcal{M}^n$ is nonsingular. For simplicity, let $C := \mathrm{Diag}(c)$ and $D := \mathrm{Diag}(d)$. We also define

$$\mathcal{Q}(c,d) := A^T \,\mathrm{Diag}(c) A \,\mathrm{Diag}(d), \;\; f(c,d) := \frac{1}{n}\,\mathrm{tr}(\mathcal{Q}(c,d)), \;\; g(c,d) := \det(\mathcal{Q}(c,d))^{1/n}. \tag{3.4}$$

Therefore we have the following equivalent problem to Equation (3.3):

$$\begin{array}{ll} \min & \omega_{\mathcal{M}}(c,d) := \frac{f(c,d)}{g(c,d)} \\ \text{s.t.} & c, d \in \mathbb{R}^n_{++}. \end{array} \tag{3.5}$$

Finally, for notational convenience, we define $\mathrm{inv} : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ by

$$\mathrm{inv}(c,d) = \begin{pmatrix} \mathrm{diag}\left(\mathrm{Diag}(c)^{-1}\right) \\ \mathrm{diag}\left(\mathrm{Diag}(d)^{-1}\right) \end{pmatrix}. \tag{3.6}$$

The following result provides the gradient of $\omega_{\mathcal{M}}(c,d)$.

Proposition 3.4. The gradient of $\omega_{\mathcal{M}}(c,d)$ is given by

$$\nabla \omega_{\mathcal{M}}(c,d) = \frac{1}{n g(c,d)} \begin{pmatrix} \mathrm{diag}(A \,\mathrm{Diag}(d) A^T) \\ \mathrm{diag}(A^T \,\mathrm{Diag}(c) A) \end{pmatrix} - \frac{1}{n} \omega_{\mathcal{M}}(c,d) \mathrm{inv}(c,d), \tag{3.7}$$

where $g(c,d)$ is as in Equation (3.4), $\omega_{\mathcal{M}}(c,d)$ is defined in Equation (3.5), and $\mathrm{inv}(c,d)$ is as in Equation (3.6).

Proof. To compute $\nabla \omega_{\mathcal{M}}(c,d)$, we have to compute $\nabla f(c,d)$ and $\nabla g(c,d)$. It is easy to see from the definition of $f(c,d)$ in Equation (3.4) that $\nabla f(c,d)$ can be computed as

$$\nabla f(c,d) = \frac{1}{n} \begin{pmatrix} \mathrm{diag}(A \,\mathrm{Diag}(d) A^T) \\ \mathrm{diag}(A^T \,\mathrm{Diag}(c) A) \end{pmatrix}. \tag{3.8}$$

To compute $\nabla g(c,d)$, observe that it follows from the definition of $g(c,d)$ in Equation (3.4) and the formula for the gradient of the determinant that

$$\begin{aligned} \langle \nabla g(c,d), (\Delta c, \Delta d) \rangle &= \det\left(A^T A\right)^{1/n} \langle \nabla \det\left(CD\right)^{1/n}, (\Delta c, \Delta d) \rangle \\ &= \det\left(A^T A\right)^{1/n} \left\langle \tfrac{1}{n} \det\left(CD\right)^{\frac{1}{n}-1} \mathrm{adj}\left(CD\right), (C\Delta D + \Delta C D) \right\rangle \\ &= \tfrac{1}{n} g(c,d) \left\langle \left(CD\right)^{-1}, (C\Delta D + \Delta C D) \right\rangle \\ &= \tfrac{1}{n} g(c,d) \,\mathrm{tr}\left(C^{-1}\Delta C + D^{-1}\Delta D\right). \end{aligned}$$

Hence, it follows from the above equations

$$\nabla g(c,d) = \frac{1}{n} g(c,d) \begin{pmatrix} \mathrm{diag}\left(\mathrm{Diag}(c)^{-1}\right) \\ \mathrm{diag}\left(\mathrm{Diag}(d)^{-1}\right) \end{pmatrix} = \frac{1}{n} g(c,d) \mathrm{inv}(c,d). \tag{3.9}$$

It then follows from Equation (3.7), Equation (3.9) and the quotient rule that $\nabla \omega_{\mathcal{M}}(c,d)$ can be computed as

$$\begin{aligned} \nabla \omega_{\mathcal{M}}(c,d) &= \frac{1}{g(c,d)^2} \left(g(c,d) \nabla f(c,d) - f(c,d) \nabla g(c,d)\right) = \frac{1}{g(c,d)} \nabla f(c,d) - \frac{\omega_{\mathcal{M}}(c,d)}{g(c,d)} \nabla g(c,d) \\ &= \frac{1}{n g(c,d)} \begin{pmatrix} \mathrm{diag}(A \,\mathrm{Diag}(d) A^T) \\ \mathrm{diag}(A^T \,\mathrm{Diag}(c) A) \end{pmatrix} - \frac{1}{n} \omega_{\mathcal{M}}(c,d) \mathrm{inv}(c,d). \end{aligned}$$

∎

Remark 3.5. It is easy to see that $\nabla \omega_{\mathcal{M}}(c,d)$ can be equivalently written in the useful form:

$$\nabla \omega_{\mathcal{M}}(c,d) = \frac{1}{n} \begin{bmatrix} 0 & A \bullet A \\ (A \bullet A)^T & 0 \end{bmatrix} \begin{pmatrix} c \\ d \end{pmatrix} - \frac{1}{n} \omega_{\mathcal{M}}(c,d) \mathrm{inv}(c,d).$$

We now state our main theorem on a necessary condition for a vector to be globally optimal for Equation (3.3). This relates to optimal matrix balancing discussed below.

THEOREM 3.6. *Let $A \in \mathbb{R}^{n\times n}$ be nonsingular. Let $c^*, d^* \in \mathbb{R}^n_{++}$ be a global optimal solution pair of Equation (3.3). Let $e_{2n} \in \mathbb{R}^{2n}$ be the vector of all ones, $C^* = \mathrm{Diag}(c^*)$, and $D^* = \mathrm{Diag}(d^*)$. Then:*

$$\begin{pmatrix} \mathrm{diag}(C^*AD^*A^T) \\ \mathrm{diag}(D^*A^TC^*A) \end{pmatrix} = \alpha e_{2n}, \quad \text{for some } \alpha > 0. \tag{3.10}$$

Hence, if c^*, d^* is a global optimal solution of Equation (3.3), then the column and row norms of $\sqrt{C^*}A\sqrt{D^*}$ must all be equal to the same constant, i.e., the matrix is balanced.

Proof. For $c^*, d^* \in \mathbb{R}^n_{++}$ to be a global optimal solution of Equation (3.3), it must satisfy the necessary condition $\nabla\omega_{\mathcal{M}}(c^*, d^*) = 0$. Thus, it follows from Equation (3.7) that c^*, d^* must satisfy

$$0 = \frac{1}{ng(c^*, d^*)} \begin{pmatrix} \mathrm{diag}(AD^*A^T) \\ \mathrm{diag}(A^TC^*A) \end{pmatrix} - \frac{1}{n}\omega_{\mathcal{M}}(c^*, d^*)\mathrm{inv}(c^*, d^*).$$

It follows from taking the Hadamard product with the vector $\begin{pmatrix} \mathrm{diag}(C^*) \\ \mathrm{diag}(D^*) \end{pmatrix}$ on both sides of the above equation as well as multiplying both sides by $ng(c^*, d^*)$ that the following necessary condition must hold

$$f(c^*, d^*)e_{2n} = \begin{pmatrix} \mathrm{diag}(C^*AD^*A^T) \\ \mathrm{diag}(D^*A^TC^*A) \end{pmatrix}. \tag{3.11}$$

∎

Observe that the definition of $f(c, d)$ in Equation (3.4) implies that if c^*, d^* satisfies the above relation then any positive scalar multiple of c^*, d^* also satisfies Equation (3.11). It then follows from this observation and Equation (3.11) that a globally optimal solution c^*, d^* must satisfy the necessary condition Equation (3.10) for some scalar $\alpha > 0$. The last observation of the theorem then immediately follows from the condition in Equation (3.10).

Remark 3.7. Suppose we consider a modification of Equation (3.3), and consider the following optimization problem:

$$\begin{array}{ll} \min & \omega((\mathrm{Diag}(c)^{1/2}A\,\mathrm{Diag}(d)^{1/2})^T(\mathrm{Diag}(c)^{1/2}A\,\mathrm{Diag}(d)^{1/2})) \\ \text{s.t.} & c \geqslant \delta e_n, d \geqslant \delta e_n, \end{array} \tag{3.12}$$

where $\delta \in (0, 1)$. It is easy to see that a global optimal solution c^*, d^* of Equation (3.12) exists. This follows immediately from the fact that $\omega\big((\mathrm{Diag}(c)^{1/2}A\,\mathrm{Diag}(d)^{1/2})^T(\mathrm{Diag}(c)^{1/2}A\,\mathrm{Diag}(d)^{1/2})\big)$ is a positively homogeneous function of degree 0, so Equation (3.12) is equivalent to minimizing a continuous function over a compact set.

3.2.2. Square-Root Sinkhorn–Knopp Algorithm for Two Sided Problem. Let a nonsingular matrix $A \in \mathcal{M}^n$ be given. We have seen above that the ω-optimal right-sided preconditioner is equivalent to the column normalization preconditioner. Similarly, the ω-optimal left-sided preconditioner is equivalent to the row normalization preconditioner. Therefore, we can alternate between ω-optimal right- and left-sided preconditioners and decrease the value of ω at each iteration. This yields the following computationally efficient matrix balancing scheme where we alternate between column and row normalization. This algorithm can be seen as equivalent to the Square-Root Sinkhorn–Knopp algorithm and is related to other balancing methods in the literature [20, 21, 28, 29].

We prove below in Theorem 3.9 that the Square-Root Sinkhorn-Knopp algorithm, namely Algorithm 3.1, converges and the outputted two-sided diagonal preconditioner satisfies the necessary condition in Equation (3.10) for global optimality of a two-sided ω-optimal preconditioner. This continues the theme that the ω-measure provides justification for heuristics used in the literature.

Algorithm 3.1 Square-Root Sinkhorn–Knopp Algorithm for Two Sided ω-Optimal Preconditioner

Inputs: A square nonsingular matrix $A \in \mathcal{M}^n$, a tolerance tol> 0, and a rule for the stopping criterion, stopcrit.

set stopcrit $\leftarrow \infty$;
set $A_0 \leftarrow A$;
set $k \leftarrow 0$;
while stopcrit$>$tol **do**
  set $k \leftarrow k+1$;
  compute $(\sqrt{d_k})_i = 1./\|(A_{k-1})_{:,i}\|$, $i = 1, \ldots n$, where $(A_{k-1})_{:,i}$ is the i-th column of A_{k-1};
  set $\tilde{A}_k \leftarrow A_{k-1}\,\mathrm{Diag}(\sqrt{d_k})$;
  compute $\sqrt{c_k} \in \mathbb{R}^n$ with $(\sqrt{c_k})_i = 1./\|(\tilde{A}_k)_{i,:}\|$, $i = 1, \ldots n$, where $(\tilde{A}_k)_{i,:}$ is the i-th row of $\tilde{A}_k$;
  update stopcrit;
  set $A_k \leftarrow \mathrm{Diag}(\sqrt{c_k})\tilde{A}_k$;
end while(main outer loop)

Output:

$$\hat{C}^{1/2} := \mathrm{Diag}\left(\prod_{j=1}^{k}\sqrt{c_j}\right) \text{ and } \hat{D}^{1/2} := \mathrm{Diag}\left(\prod_{j=1}^{k}\sqrt{d_j}\right).$$

There several viable options for the choice of stopcrit in Algorithm 3.1. One natural choice is

$$\text{stopcrit} = \max\left(\max_i \left|\|(A_k)_{i,:}\| - 1\right|, \max_j \left|\|(A_k)_{:,j}\| - 1\right|\right).$$

It is shown in Theorem 3.9 that the sequence A_k converges to a matrix that is balanced, i.e., one that has column and row norms all equal to one. The remark below also discusses that the ω values of the iterates are monotonically decreasing.

Remark 3.8. At each iteration, $\tilde{A}_k$ (resp. A_k) is computed using the ω-optimal right-sided (resp. left-sided) preconditioner. Hence, the ω values of the iterates are monotonically decreasing, i.e., $\omega(A_k A_k^T) \leqslant \omega(\tilde{A}_k \tilde{A}_k^T) \leqslant \omega(A_{k-1} x A_{k-1}^T)$ for all $k \geqslant 1$. The output of Algorithm 3.1 thus satisfies $\omega((\hat{C}^{1/2} A \hat{D}^{1/2})^T)(\hat{C}^{1/2} A \hat{D}^{1/2})) < \omega(A^T A)$.

Theorem 3.9 now provides a convergence guarantee for Algorithm 3.1 and shows that the necessary condition Equation (3.10) for a two-sided ω-optimal diagonal preconditioner holds in the limit for Algorithm 3.1.

THEOREM 3.9. *Let A be such that $A \bullet A$ has total support.*[9] *Then the sequence A_k converges linearly to a matrix $\bar{A} := \bar{C}^{1/2} A \bar{D}^{1/2}$ that has column and row norms all equal to 1. That is,*

$$\lim_{k\to\infty} \mathrm{Diag}\left(\prod_{j=1}^{k}\sqrt{c_j}\right) A\, \mathrm{Diag}\left(\prod_{j=1}^{k}\sqrt{d_j}\right) = \bar{C}^{1/2} A \bar{D}^{1/2}$$

with a linear rate of convergence. Hence, it follows from Theorem 3.6 that limiting matrices $\bar{C}$ and $\bar{D}$ satisfy the necessary condition Equation (3.10) for a two-sided ω-optimal diagonal preconditioner.

Proof. Let $\tilde{P}_k = \tilde{A}_k \bullet \tilde{A}_k$ and let $P_k = A_k \bullet A_k$. It follows that $\tilde{P}_k = (A_{k-1} \bullet A_{k-1})\,\mathrm{Diag}(d_k) = P_{k-1}\,\mathrm{Diag}(d_k)$ where $(d_k)_i = 1./\|(A_{k-1})_{:,i}\|^2$. Note $\|(A_{k-1})_{:,i}\|^2$ is exactly the i-th column sum of P_{k-1} so $(d_k)_i = 1./\sum_{l=1}^n (P_{k-1})_{l,i}$. Moreover, also observe that $P_k = \mathrm{Diag}(c_k)(\tilde{A}_k \bullet \tilde{A}_k) = \mathrm{Diag}(c_k)\tilde{P}_k$. Note $(c_k)_i = 1./\sum_{l=1}^n (\tilde{P}_k)_{i,l}$. Hence, the sequences of iterates $\tilde{P}_k$ and P_k can be viewed as the iterates of Sinkhorn–Knopp Algorithm applied to $A \bullet A$. By [20], we know then that the sequence P_k converges linearly to a matrix $\bar{P}$ where $\bar{P} := \bar{C}(A \bullet A)\bar{D}$ is a doubly stochastic matrix. The sequences of iterates $\tilde{A}_k$ and A_k are just the square roots of the iterates of the Sinkhorn–Knopp algorithm applied to $A \bullet A$. Therefore, it follows from the above argument that the sequence A_k converges linearly to a matrix $\bar{A} := \bar{C}^{1/2} A \bar{D}^{1/2}$ that has column and row norms all equal to 1. The last conclusion of the theorem follows from this observation and the necessary condition Equation (3.10) in Theorem 3.6.

[9] A nonnegative square matrix B is said to have total support if $B \neq 0$ and all its nonzero elements lie on a positive diagonal. In this case, a diagonal of a matrix is just a collection of elements with one element from each row and one element from each column of B. More specifically, it will consist of $b_{i,\sigma(i)}$ for $i = 1, 2, \ldots n$ for a permutation σ and where b_{ij} denotes the (i, j) entry of B.

3.3. Right-, Left-Sided Block Diagonal. We now show that the above results extend directly to finding *block diagonal* preconditioners for least squares solutions of full rank possibly overdetermined linear systems $Ax = b, A \in \mathbb{R}^{m\times n}, m \geqslant n$. The preconditioner is extended from diagonal to block diagonal (block upper-triangular). This relates to sparse QR preconditioning and extends the results for ω-optimal preconditioning with partial Cholesky structure in [17, Theorem 2.7] and the block diagonal ω-optimal preconditioner in [8, Prop. 3 part 3].

3.3.1. Right-Sided Block Diagonal. We let the linear transformation $B = \text{Blkdiag}(\mathcal{B}) \in \mathcal{M}^n$ denote the block diagonal matrix with blocks formed from the set of square matrices $\mathcal{B} = \{B_i\}_{i=1}^k$ of order $n_i, \sum_{i=1}^k n_i = n$. For our application we restrict the blocks to be of upper-triangular structure denoted $\mathcal{R} = \{R_i\}_{i=1}^k$, and see that the best, with respect to the ω measure, comes from Q-less QR decompositions of the corresponding blocks of A.

First we choose the number of blocks k and the block sizes $n_i, \sum_{i=1}^k n_i = n$. Thus we get the block structure

$$A = \begin{bmatrix} A_1 & A_2 & \dots & A_k \end{bmatrix}, \quad A/\,\text{Blkdiag}(\mathcal{R}) = \begin{bmatrix} A_1/R_1 & A_2/R_2 & \dots & A_k/R_k \end{bmatrix},$$

where we use the MATLAB notation that / denotes matrix division $A/R = AR^{-1}$. Optimally, we want the sizes of the blocks chosen so that the matrices $A_i^T A_i$ are chordal so that there is no loss of sparsity. Moreover, we can allow a preliminary permutation of the columns $A \leftarrow AP$ so that we can increase the effect of the preconditioner.

To extend the diagonal preconditioner results we solve:

$$\min_{\mathcal{B}} \{\omega((AB)^T(AB)) : B = \text{Blkdiag}(\mathcal{B})\}.$$

The basic result we use follows.

Proposition 3.10 ([8, Prop. 3 part 3], [9, Prop. 2.2]). Let

$$A = \begin{bmatrix} A_1 & A_2 & \dots & A_k \end{bmatrix}, \quad A_i \in \mathbb{R}^{m\times n_i}, \forall i,$$

be a full rank $m \times n$ matrix, $m \geqslant n$. Then an optimal block diagonal scaling

$$\mathcal{B} := \{B_1, B_2, \dots, B_k\},\ B_i \in \mathbb{R}^{n_i \times n_i}, \qquad B := \text{Blkdiag}(\mathcal{B}),$$

that minimizes the measure ω, i.e.,

$$\min \{\omega((AB)^T(AB)) : B = \text{Blkdiag}(\mathcal{B})\},$$

is found by satisfying the factorization

$$B_i B_i^T = (A_i^T A_i)^{-1}, \quad i = 1, \dots, k.$$

COROLLARY 3.11. *Let $A, \mathcal{B}, B$ be as in Proposition 3.10.*
(i) If we restrict the diagonal blocks B_i to be diagonal themselves, then we get the optimal diagonal preconditioner in Proposition 3.1.
(ii) If we restrict the diagonal blocks B_i to be upper-triangular, then we get $B_i = R_i^{-1}, A_i = Q_i R_i$ from the QR decomposition of A_i (up to sign of rows of R).

3.3.2. Left-Sided Block Diagonal. We continue with A full column rank but with left-sided preconditioning. We present a nonlinear equation that characterizes optimality.

Theorem 3.12. Let $A \in \mathbb{R}^{m\times n}$ be full column rank with block structure given by

$$A = \begin{bmatrix} \bar{A}_1 \\ \bar{A}_2 \\ \dots \\ \bar{A}_\ell \end{bmatrix}, \bar{A}_j \in \mathbb{R}^{m_j \times n}, \text{rank}(\bar{A}_j) = m_j,\ \forall j.$$

Then the ω-optimal left-sided block diagonal preconditioner

$$E := \text{Blkdiag}(E_1, E_2, \dots, E_\ell),\ E_j \in \mathbb{R}^{m_j \times m_j},$$

that minimizes the measure ω, i.e.,

$$\min \omega((EA)^T EA),$$

is characterized by:

$$\bar{A}_j\bar{A}_j^T = \frac{1}{n}\left(\operatorname{tr}\sum_{t=1}^{\ell}\left(\bar{A}_t^T(E_t^TE_t)\bar{A}_t\right)\right)\bar{A}_j\left(A^T(E^TE)A\right)^{-1}\bar{A}_j^T,\ \forall j=1,\dots\ell.$$

Proof. To begin, we substitute $K := E^TE$. That is, we solve

$$\min\{\omega(A^TKA) : K = \operatorname{Blkdiag}(K_1,K_2,\dots,K_\ell),\ K_j\in\mathbb{R}^{m_j\times m_j}\}.$$

Notice that $A^TKA = \sum_{j=1}^{\ell}\bar{A}_j^TK_j\bar{A}_j$. Define the linear transformation $\mathcal{G}(K) = \sum_{j=1}^{\ell}\bar{A}_j^TK_j\bar{A}_j$. This yields

$$\omega_{\mathcal{G}}(K) := \frac{\operatorname{tr}(A^TKA)/n}{\det(A^TKA)^{1/n}} = \frac{\sum_{j=1}^{\ell}\operatorname{tr}(\bar{A}_j^TK_j\bar{A}_j)/n}{\det(A^TKA)^{1/n}}.$$

Hence, for convenience and for the remainder of the proof define

$$f(K) := \sum_{j=1}^{\ell}\operatorname{tr}(\bar{A}_j^TK_j\bar{A}_j)/n \quad\text{and}\quad g(K) := \det\left(A^TKA\right)^{1/n}.$$

The partial derivative of g with respect to the block K_j is given by

$$\begin{aligned}\langle\nabla_{K_j}g(K),\Delta K_j\rangle &= \langle\nabla\det(\mathcal{G}(K))^{1/n},\Delta K_j\rangle\\ &= \langle\frac{1}{n}\det(\mathcal{G}(K))^{1/n-1}\operatorname{adj}(\mathcal{G}(K)),\bar{A}_j^T\Delta K_j\bar{A}_j\rangle\\ &= \frac{1}{n}g(K)\langle\mathcal{G}(K)^{-1},\bar{A}_j^T\Delta K_j\bar{A}_j\rangle\\ &= \frac{1}{n}g(K)\langle\bar{A}_j\mathcal{G}(K)^{-1}\bar{A}_j^T,\Delta K_j\rangle.\end{aligned}$$

Therefore, we have

$$\nabla_{K_j}f(K) = \frac{1}{n}\bar{A}_j\bar{A}_j^T \quad\text{and}\quad \nabla_{K_j}g(K) = \frac{1}{n}g(K)\bar{A}_j(A^TKA)^{-1}\bar{A}_j.$$

These in turn lead to the partial derivative of $\omega_{\mathcal{G}}$ with respect to K_j given by

$$\begin{aligned}\nabla_{K_j}\omega_{\mathcal{G}}(K) &= \frac{1}{g(K)^2}(g(K)\nabla_{K_j}f(K) - f(K)\nabla_{K_j}g(K))\\ &= \frac{1}{ng(K)}(\bar{A}_j\bar{A}_j^T - f(K)\bar{A}_j(A^TKA)^{-1}\bar{A}_j^T)\end{aligned}$$

Therefore

$$\nabla\omega_{\mathcal{G}}(K) = 0 \iff \bar{A}_j\bar{A}_j^T = \frac{1}{n}\sum_{t=1}^{\ell}\operatorname{tr}(\bar{A}_t\bar{A}_t^TK_t)\,\bar{A}_j(A^TKA)^{-1}\bar{A}_j^T,\quad\forall j.$$

∎

We can then recover the E_j from K using factorizations of the blocks $K_j = E_j^TE_j$.

Finding an explicit solution for E in Theorem 3.12 is an open question. However, if we assume further that A is a square matrix, the following Corollary 3.13 connects the result from Corollary 3.11 to this left sided preconditioning.

Corollary 3.13. Let $A\in\mathcal{M}^n$ be nonsingular. Let the block of rows, $\bar{A}_j\in\mathbb{R}^{n_j\times n}$, be given by $A^T = [\bar{A}_1^T\ \bar{A}_2^T\ \dots\ \bar{A}_\ell^T]$, $\sum_j n_j = n$. Define $\mathcal{E} := \{E_1,E_2,\dots,E_k\}$ by $E_i := \bar{R}_i^{-T}$, where $\bar{A}_i\bar{A}_i^T := \bar{R}_i^T\bar{R}_i$ is the Cholesky decomposition. Then, $\mathcal{E}$ is the optimal lower triangular block diagonal scaling for

$$\min\{\omega((EA)^T(EA)) : E = \operatorname{Blkdiag}(E_1,E_2,\dots,E_k)\}.$$

Proof. Since A and E are square matrices, we observe

$$\omega\left((EA)^TEA\right) = \omega\left(A^TE^TEA\right) = \omega\left(EAA^TE^T\right),$$

and thus Corollary 3.11 implies $E_i = \bar{R}_i^{-T}$, where $\bar{A}_i\bar{A}_i^T := \bar{R}_i^T\bar{R}_i$ as defined above. ∎

Remark 3.14. The result in Corollary 3.13 agrees with our characterization in Theorem 3.12. Indeed, by defining $E_i := \bar{R}_i^{-T}$ with $\bar{A}_i\bar{A}_i^T = \bar{R}_i^T\bar{R}_i$, we get that $\operatorname{tr}(\bar{A}_t\bar{A}_t^T E_t^T E_t) = \operatorname{tr}(\bar{R}_t^T\bar{R}_t\bar{R}_t^{-1}\bar{R}_t^{-T}) = \operatorname{tr}(I_{n_t}) = n_t$. Thus,

$$\sum_{t=1}^{\ell} \operatorname{tr}(\bar{A}_t\bar{A}_t^T E_t^T E_t) = \sum_{t=1}^{\ell} n_t = n.$$

Now, define $\mathcal{I}_j := \begin{bmatrix} 0 & I_{n_j} & 0 \end{bmatrix} \in \mathbb{R}^{n_j \times n}$, with the identity $I_{n_j} \in \mathcal{M}^{n_j}$ of order n_j, appropriately located such that $\mathcal{I}_j A = \bar{A}_j$. Then,

$$\begin{aligned} \frac{1}{n}\left(\operatorname{tr}\sum_{t=1}^{\ell}\left(\bar{A}_t^T E_t^T E_t \bar{A}_t\right)\right)\bar{A}_j\left(A^T E^T E A\right)^{-1}\bar{A}_j^T &= \bar{A}_j(A^T E^T E A)^{-1}\bar{A}_j^T \\ &= \mathcal{I}_j A A^{-1}E^{-1}E^{-T}A^{-T}A^T\mathcal{I}_j^T \\ &= \mathcal{I}_j E^{-1}E^{-T}\mathcal{I}_j^T \\ &= E_j^{-1}E_j^{-T} = \bar{R}_j^T\bar{R}_j = \bar{A}_j\bar{A}_j^T. \end{aligned}$$

4. Computational Experiments. Our computational experiments are divided into several parts. In Subsection 4.1, we compare the computational efficiency of our subgradient algorithms, Algorithms 2.2 and B.1, with the SDP-based approaches proposed in [25] and [12] for finding an approximate κ-optimal preconditioner. Both our codes and the code in [12] take as input $M \succ 0$. Our subgradient algorithms find a diagonal D that minimizes $\kappa(D^{1/2}MD^{1/2})$ while [12] solves an SDP using MOSEK [24] to find a D that is κ-optimal over the subspace spanned by a small list of given diagonal preconditioners. Meanwhile, [25] obtains a diagonal scaling E that minimizes $\kappa((ME)^T(ME))$ by solving a possibly larger SDP. Hence, when using their code, we input a Cholesky factor B of M, i.e., $M = B^TB$, so that their algorithm finds E that minimizes $\kappa(EME)$. For every experiment, $\kappa(M)$, $\kappa(D^{1/2}MD^{1/2})$, and $\kappa(EME)$ were each evaluated by performing a minimum and maximum eigenvalue computation using MATLAB's eigs function with 10^{-10} precision.

In Subsection 4.2, we compare PCG using the preconditioner from Algorithm 2.2 with PCG using the preconditioner of [25] for solving $Mx = b$, where $M \succ 0$. In Subsection 4.3, we compare the efficiency of our Algorithm 3.1, with the two-sided κ-optimal preconditioner presented in [25]. In particular, we compare the time to compute each preconditioner as well as the performance of each preconditioner when used inside LSQR to minimize $\|b - Ax\|$ where $A \in \mathcal{M}^n$ is nonsingular. Finally, in Subsection 4.4, we use the optimality conditions in Theorem 2.7 in order to construct a matrix M that is κ-optimal with respect to diagonal preconditioning. We then compare with results after applying the Jacobi scaling, the ω-optimal scaling.

All experiments in Subsections 4.1 to 4.3 are run on a 2023 Macbook Pro with an 8-core CPU and 128 GB of memory using MATLAB 2024a. The medium experiments in Subsection 4.4 were also run with this Macbook Pro while the very large instances were run with a large Linux machine from University of Waterloo with 256 GB of memory. The latest codes are available at this clickable-link or with URL https://github.com/asujanani6/Optimal_Preconditioning.

4.1. Minimizing κ Efficiently. In this subsection, Tables 4.1 and 4.2 compare the computational efficiency of Algorithm 2.2 with the algorithms in [25] and [12] for minimizing κ. The tables also compare these algorithms with our own line-search based subgradient method, namely Algorithm B.1, that was slightly more computationally efficient than our Algorithm 2.2 although Algorithm 2.2 has theoretical convergence guarantees. The precise details of Algorithm B.1 can be found in Appendix B. In our experiments, Algorithm 2.2 uses a tol of 10^{-4}, sets stopcrit=$2|\kappa(d_{k+1}) - \kappa(d_k)|/|\kappa(d_{k+1})) + \kappa(d_k)|$ at every iteration, and uses a maxiter of 500 and $\hat{\delta} = 10^{-3}$. The default settings for the algorithms in [25] and [12] are used. Algorithm B.1 uses a maximum of 80 iterations and uses the same tolerance 10^{-4}.

We now give several remarks about the results presented in each table. Table 4.1 compares the four algorithms on eighteen matrices taken from the SuiteSparse Matrix Collection [6]. The list of the SuiteSparse matrices considered can be found in Appendix C. Columns four to seven of Table 4.1 list the percent reduction in κ that each method achieved. The percentage reduction is computed as $100*(\kappa(M)-\kappa(\hat{M}))/(\kappa(M))$, where $\hat{M}$ is $D^{1/2}MD^{1/2}$ for our codes and [12] or EME for [25]. As seen from these columns, both Algorithms 2.2 and B.1 reduced κ much more significantly than [25] and [12]. Algorithm 2.2 reduced κ on average by 79.7% while Algorithm B.1 reduced κ by 68.5%. On the other hand, [25] reduced κ on average by 30.3% while [12]

TABLE 4.1
Comparison of Algorithms for Min κ on SuiteSparse

Dim & Density, & $\kappa(M)$			% Reduction in κ				CPU Time (seconds)				CPU Ratios	
n	density	$\kappa(M)$	[25]	[12]	Algorithm 2.2	Algorithm B.1	[25]	[12]	Algorithm 2.2	Algorithm B.1	[25]/Algorithm 2.2	[25]/Algorithm B.1
1074	1.1e-02	2.6e+07	0.0e+00	0.0e+00	9.9e+01	9.9e+01	48.203	51.700	8.469	5.644	5.7	8.5
2003	2.1e-02	1.1e+10	0.0e+00	0.0e+00	9.8e+01	6.8e+01	330.887	324.334	12.968	0.120	25.5	2752.3
3600	2.1e-03	1.8e+07	0.0e+00	0.0e+00	6.0e+01	6.0e+01	510.244	1102.270	24.644	9.538	20.7	53.5
3134	4.6e-03	2.6e+12	-9.7e-04	0.0e+00	7.5e+01	7.0e+01	178.360	1347.551	2.542	0.410	70.2	435.3
3562	1.3e-02	1.9e+11	0.0e+00	0.0e+00	8.6e+01	7.8e+01	1621.648	948.059	36.940	1.694	43.9	957.4
1922	8.2e-03	1.7e+08	0.0e+00	0.0e+00	9.6e+01	8.8e+01	182.146	448.809	4.524	0.405	40.3	450.0
4410	1.1e-02	9.5e+08	0.0e+00	8.7e+01	9.6e+01	8.3e+01	3170.182	2602.683	4.481	2.784	707.5	1138.7
588	6.2e-02	2.8e+04	9.9e+01	0.0e+00	9.0e+01	9.4e+01	19.195	7.152	0.463	1.379	41.5	13.9
494	6.8e-03	2.4e+06	9.0e+01	0.0e+00	9.1e+01	3.3e+01	9.222	2.959	0.832	0.116	11.1	79.8
662	5.6e-03	7.9e+05	9.5e+01	0.0e+00	7.4e+01	4.5e+01	16.293	5.947	0.324	0.203	50.2	80.3
1824	1.2e-02	1.9e+06	0.0e+00	0.0e+00	8.7e+01	7.1e+01	182.642	165.698	3.468	1.958	52.7	93.3
2146	1.6e-02	1.7e+03	0.0e+00	5.3e+01	5.5e+01	4.9e+01	428.711	137.240	2.604	16.508	164.6	26.0
2910	2.1e-02	6.0e+06	0.0e+00	0.0e+00	9.1e+01	7.5e+01	619.758	613.440	8.306	3.731	74.6	166.1
237	1.8e-02	2.0e+07	2.1e+01	5.5e+00	7.9e+01	6.5e+01	2.178	0.620	0.276	0.056	7.9	38.8
957	4.5e-03	5.1e+09	2.8e+01	3.5e+01	6.0e+01	6.5e+01	10.107	8.676	6.419	0.145	1.6	69.7
100	5.9e-02	1.6e+03	3.8e+01	3.1e+01	3.5e+01	3.5e+01	0.800	0.297	0.036	0.970	22.2	0.8
468	2.4e-02	1.1e+04	8.3e+01	7.9e+01	7.7e+01	7.1e+01	15.428	2.957	1.081	1.298	14.3	11.9
729	8.7e-03	2.4e+09	9.2e+01	8.9e+01	8.6e+01	8.4e+01	39.553	6.439	0.987	1.108	40.1	35.7

TABLE 4.2
Comparison of Algorithms for Min κ on Random

Dim & Density, & $\kappa(A)$			% Reduction in κ				CPU Time (seconds)				CPU Ratios	
n	density	$\kappa(M)$	[25]	[12]	Algorithm 2.2	Algorithm B.1	[25]	[12]	Algorithm 2.2	Algorithm B.1	[25]/Algorithm 2.2	[25]/Algorithm B.1
2000	1.0e-03	2.0e+06	-4.8e+01	4.8e+01	5.3e+01	6.6e+01	14.498	76.446	4.470	1.154	3.2	12.6
2500	8.1e-04	2.5e+06	-5.0e+01	2.9e+01	5.2e+01	5.2e+01	21.500	229.744	9.068	1.233	2.4	17.4
3000	6.7e-04	3.0e+06	-1.8e+01	0.0e+00	1.9e+01	1.3e+01	32.757	358.866	5.011	0.476	6.5	68.8
3500	5.8e-04	3.5e+06	-1.9e+01	2.5e+01	4.3e+01	4.9e+01	43.032	565.527	8.757	1.961	4.9	21.9
4000	5.0e-04	4.0e+06	-4.4e+01	0.0e+00	2.8e+01	6.3e+00	56.246	631.303	11.281	0.519	5.0	108.4
4500	4.5e-04	4.5e+06	-3.0e+01	0.0e+00	3.3e+01	4.2e+01	69.000	888.172	3.774	2.202	18.3	31.3
5000	4.0e-04	5.0e+06	-4.4e+01	0.0e+00	3.9e+01	3.9e+01	85.069	2009.496	13.994	2.341	6.1	36.3
5500	3.7e-04	5.5e+06	-3.5e+01	1.4e+01	2.8e+01	2.8e+01	104.179	2512.849	2.763	2.614	37.7	39.8
6000	3.3e-04	6.0e+06	-4.3e+01	0.0e+00	3.0e+01	3.4e+01	127.060	3680.562	5.383	2.748	23.6	46.2
6500	3.1e-04	6.5e+06	-2.9e+01	1.9e+01	2.7e+01	3.2e+01	146.476	3755.564	4.694	3.124	31.2	46.9
7000	2.9e-04	7.0e+06	-2.9e+01	2.4e+01	3.3e+01	3.0e+01	168.726	3838.763	11.441	3.243	14.7	52.0
7500	2.7e-04	7.5e+06	-4.1e+01	2.5e+01	3.2e+01	2.8e+01	193.993	3879.417	12.929	3.649	15.0	53.2
8000	2.5e-04	8.0e+06	-1.9e+01	3.0e+01	3.2e+01	2.7e+01	215.347	3790.944	16.234	3.645	13.3	59.1
8500	2.4e-04	8.5e+06	-2.8e+01	3.6e+01	2.5e+01	2.6e+01	246.213	3929.044	8.086	4.716	30.5	52.2
9000	2.2e-04	9.0e+06	-1.5e+01	-9.9e+00	9.1e+00	3.2e+00	271.863	3903.914	7.354	0.931	37.0	292.1
9500	2.1e-04	9.5e+06	-4.2e+01	3.6e+01	2.6e+01	2.3e+01	323.748	4147.347	13.329	5.681	24.3	57.0
10000	2.0e-04	1.0e+07	-3.2e+01	1.2e+01	2.2e+01	1.8e+01	341.504	4304.042	8.240	5.232	41.4	65.3
10500	1.9e-04	1.0e+07	-1.2e+01	0.0e+00	2.7e+01	2.1e+01	384.690	4302.243	20.255	6.627	19.0	58.0
11000	1.8e-04	1.1e+07	-4.1e+01	3.0e+00	2.2e+01	1.9e+01	463.158	4273.218	11.945	6.085	38.8	76.1
11500	1.7e-04	1.1e+07	-3.4e+01	0.0e+00	2.3e+01	2.0e+01	487.216	4000.550	9.915	6.705	49.1	72.7
12000	1.7e-04	1.2e+07	-3.0e+01	0.0e+00	2.2e+01	1.9e+01	511.686	4581.828	10.963	6.709	46.7	76.3
12500	1.6e-04	1.3e+07	-3.1e+01	0.0e+00	4.0e+00	3.2e+00	546.856	4059.532	3.496	2.080	156.4	263.0
13000	1.5e-04	1.3e+07	-2.9e+01	0.0e+00	2.4e+01	1.8e+01	593.252	4586.045	40.099	7.383	14.8	80.4
13500	1.5e-04	1.4e+07	-3.0e+01	0.0e+00	2.2e+01	1.8e+01	665.554	5145.811	15.941	8.300	41.8	80.2
14000	1.4e-04	1.4e+07	-3.0e+01	0.0e+00	2.2e+01	1.7e+01	719.336	4122.784	14.430	7.864	49.9	91.5
14500	1.4e-04	1.4e+07	-3.4e+01	0.0e+00	2.1e+01	1.2e+01	901.989	4606.777	24.572	6.888	36.7	130.9
15000	1.3e-04	1.5e+07	-4.1e+01	0.0e+00	1.6e+01	1.1e+01	920.641	5115.021	14.827	6.354	62.1	144.9

reduced κ by 21.08%. Columns eight to eleven of Table 4.1 show that Algorithms 2.2 and B.1 were also much faster in reducing κ than [25] and [12]. On average, Algorithm 2.2 was 77.5 (resp. 85.80) times faster than [25] (resp. [12]) while Algorithm B.1 was 356.2 (resp. 506.31) times faster than [25] (resp. [12]).

Similar results are presented in Table 4.2, which considers matrices that were randomly generated to have a specified reciprocal condition number using MATLAB's sprandsym function. For these instances, we impose a time limit for the algorithm proposed in [12], as these problems were more computationally demanding for their code. We set the time limit to 3600 seconds, although in some cases, their code terminates slightly after this limit.

On every instance considered in Table 4.2, Algorithm 2.2 and Algorithm B.1 reduced κ more than [25] and [12] and were also much more efficient than these two algorithms. Algorithm 2.2 and Algorithm B.1 reduced κ on every instance while [25] (resp. [12]) reduced κ on 0 (resp. 12) of the 27 instances considered. Moreover, Algorithm 2.2 (resp. Algorithm B.1) was also over 15 times faster than [25] and [12] on 18 (resp. 26) of the 27 instances considered.

Tables 4.3 and 4.4 illustrate the scalability of our Algorithms 2.2 and B.1 in efficiently minimizing κ on large test instances. We do not compare against [25] and [12] since both codes were not able to handle such large instances in a reasonable amount of time. Table 4.3 considers matrices that were taken from the SuiteSparse Matrix Collection while Table 4.4 considers matrices that were randomly generated using sprandsym to have very large κ values. The results in Table 4.3 show that on average, Algorithm 2.2 reduced κ by 45.7% while Algorithm B.1 reduced κ by 41.9%. Both algorithms also frequently took less than one minute to minimize κ. Table 4.4 shows that, across eleven difficult instances with $\kappa(M) > 10^{11}$, Algorithm 2.2 reduced κ by an average of 11.6%, while Algorithm B.1 achieved a 4.4% reduction.

4.2. PCG Comparison for Solving Linear Systems. In this subsection, we assume $M > 0$ and compare the performance of PCG for solving $Mx = b$ using the preconditioner from Algorithm B.1 with the one obtained by [25]. PCG is run with a tolerance of $1e-4$ and a maximum iteration count of $5e6$. The results are presented in Table 4.5. All matrices M considered in both tables are randomly generated, using MATLAB's sprandsym function, with varying dimensions, densities, and κ values.

We see in Table 4.5 that on average, across the 47 instances in Table 4.5, PCG takes approximately

TABLE 4.3
Comparison of Algorithms for Min κ on Large SuiteSparse

Dim & Density, & $\kappa(M)$			% Reduction in κ		CPU Time	
n	density	$\kappa(M)$	Algorithm 2.2	Algorithm B.1	Algorithm 2.2	Algorithm B.1
14822	3.3e-03	2.0e+06	3.9e+01	3.0e+01	7.850	283.641
15439	1.1e-03	4.4e+12	8.0e+01	7.2e+01	8.734	2.185
15439	6.5e-05	6.1e+09	4.2e+01	5.4e+01	2.283	2.725
17361	1.1e-03	2.5e+06	1.3e+01	1.2e+01	1.097	9.086
17361	3.4e-03	1.1e+09	5.6e+01	3.0e+01	19.675	9.306
23052	2.2e-03	7.4e+11	9.0e+01	7.2e+01	503.668	6.933
30401	5.1e-04	5.8e+03	9.1e+01	9.3e+01	204.718	14.951
36441	4.3e-04	2.6e+03	8.5e+01	9.2e+01	459.869	59.632
40806	1.2e-04	8.1e+01	4.5e-02	1.1e-01	29.901	16.301
48962	2.1e-04	1.6e+05	6.2e+00	4.1e+00	22.109	43.123
150102	3.2e-05	1.3e+07	4.6e-03	1.5e+00	53.450	864.384

TABLE 4.4
Comparison of Algorithms for Min κ on Large Random

Dim & Density, & $\kappa(M)$			% Reduction in κ		CPU Time	
n	density	$\kappa(M)$	Algorithm 2.2	Algorithm B.1	Algorithm 2.2	Algorithm B.1
50000	3.8e-05	1.0e+11	1.7e+01	7.7e+00	244.577	74.798
60000	3.2e-05	1.2e+11	1.5e+01	6.5e+00	400.467	103.404
70000	2.7e-05	1.4e+11	1.4e+01	5.6e+00	492.698	123.284
80000	2.4e-05	1.6e+11	1.1e+01	5.0e+00	436.111	198.894
90000	2.1e-05	1.8e+11	1.0e+01	4.3e+00	550.245	223.299
100000	1.9e-05	2.0e+11	1.2e+01	4.0e+00	986.187	250.367
110000	1.7e-05	2.2e+11	1.1e+01	3.7e+00	1162.321	293.284
120000	1.6e-05	2.4e+11	1.1e+01	3.4e+00	1172.119	332.887
130000	1.5e-05	2.6e+11	8.5e+00	3.1e+00	1274.404	386.352
140000	1.4e-05	2.8e+11	9.1e+00	2.9e+00	1500.467	480.042
150000	1.3e-05	3.0e+11	8.7e+00	2.7e+00	1677.360	498.512

442,147 iterations using the preconditioner obtained from Algorithm B.1 while it takes 549,002 iterations using the preconditioner from [25]. Moreover, the average total CPU time (including time for both preconditioning and PCG) for Algorithm B.1 is 34.0 seconds, compared to 334.1 seconds for the algorithm in [25]. Hence, in terms of total CPU time, Algorithm B.1 is on average approximately ten times faster than the method in [25].

TABLE 4.5
PCG Comparison Using Preconditioners Found by Algorithm in [25] and Algorithm B.1

Dim, Density, & $\kappa(A)$			% Reduction in κ		PCG Iterations		PCG CPU Time		Total CPU Time	
n	density	$\kappa(M)$	[25]	Algorithm B.1	[25]	Algorithm B.1	[25]	Algorithm B.1	[25]	Algorithm B.1
1000	3.3e-03	1.0e+09	-2.6e+01	3.6e+01	115264	99023	1.14	0.99	5.33	1.29
1300	2.5e-03	1.3e+09	-2.6e+01	6.0e+01	153760	101197	1.89	1.23	8.41	1.49
1600	2.0e-03	1.6e+09	-4.3e+01	-1.0e-13	180929	179469	2.67	2.67	12.17	2.71
1900	1.7e-03	1.9e+09	-3.5e+01	6.9e+01	202276	121629	3.43	2.07	16.17	2.87
2200	1.4e-03	2.2e+09	-4.9e+01	7.1e+01	245521	126567	4.82	2.50	21.54	3.39
2500	1.2e-03	2.5e+09	-2.3e+01	2.7e+01	252757	221181	5.43	4.69	29.06	5.07
2800	1.1e-03	2.8e+09	-4.2e+01	1.9e+01	289351	248376	6.69	5.74	33.60	6.12
3100	9.9e-04	3.1e+09	-2.9e+01	6.6e+01	306195	174755	8.02	4.56	45.28	6.01
3400	9.0e-04	3.4e+09	-3.3e+01	5.4e+01	325838	215488	9.07	6.05	54.35	7.46
3700	8.3e-04	3.7e+09	-2.5e+01	4.4e+01	330583	256787	9.89	7.67	63.61	9.21
4000	7.6e-04	4.0e+09	-4.4e+01	5.7e+01	366269	234646	11.60	7.40	65.77	9.21
4300	7.1e-04	4.3e+09	-2.9e+01	5.4e+01	378370	252209	13.49	9.00	88.27	10.97
4600	6.6e-04	4.6e+09	-4.2e+01	4.1e-13	408179	384608	15.52	14.65	89.55	14.74
4900	6.2e-04	4.9e+09	-1.1e+01	8.7e+00	399371	382842	15.87	15.31	108.88	15.75
5200	5.8e-04	5.2e+09	-3.7e+01	-3.7e-14	458497	418282	19.04	17.32	115.01	17.43
5500	5.5e-04	5.5e+09	-3.3e+01	3.9e+01	460157	340511	19.97	14.79	128.37	16.99
5800	5.2e-04	5.8e+09	-3.2e+01	4.6e+01	469674	336957	21.25	15.32	143.80	17.73
6100	4.9e-04	6.1e+09	-5.0e+01	4.3e+01	506736	353612	23.89	16.68	157.42	19.34
6400	4.7e-04	6.4e+09	-2.2e+01	4.3e+01	492739	361695	23.98	17.59	166.18	20.56
6700	4.4e-04	6.7e+09	-4.0e+01	4.1e+01	526574	378065	25.65	18.54	189.81	21.40
7000	4.2e-04	7.0e+09	-2.9e+01	3.9e+01	533440	394622	28.16	20.13	202.50	22.93
7300	4.1e-04	7.3e+09	-5.3e+01	2.7e+00	572826	511612	32.32	27.95	234.05	28.40
7600	3.9e-04	7.6e+09	-2.9e+01	-3.8e-14	555453	528584	31.27	29.77	232.71	29.89
7900	3.7e-04	7.9e+09	-2.2e+01	3.5e+01	569653	434589	33.24	25.44	274.74	29.14
8200	3.6e-04	8.2e+09	-6.6e+01	3.5e+01	614529	448278	37.09	26.97	323.07	30.42
8500	3.5e-04	8.5e+09	-2.7e+01	3.4e+01	573590	458996	35.49	28.28	292.58	32.18
8800	3.3e-04	8.8e+09	-3.6e+01	1.6e+01	625430	532398	40.04	34.12	322.79	35.85
9100	3.2e-04	9.1e+09	-1.8e+01	1.5e+01	597815	541439	39.21	35.60	324.73	37.50
9400	3.1e-04	9.4e+09	-2.0e+01	1.1e+01	635767	565196	42.77	37.92	374.03	39.77
9700	3.0e-04	9.7e+09	-2.9e+01	3.1e+01	639683	508698	44.28	35.17	391.15	40.05
10000	2.9e-04	1.0e+10	-4.5e+01	3.1e+01	687294	521357	48.91	37.36	454.89	41.62
10300	2.8e-04	1.0e+10	-2.5e+01	2.9e+01	667536	538749	49.08	39.58	506.66	44.78
10600	2.8e-04	1.1e+10	-4.4e+01	2.9e+01	725311	548238	54.63	41.32	510.40	46.14
10900	2.7e-04	1.1e+10	-3.9e+01	2.6e+01	725604	566231	55.80	43.64	491.97	49.26
11200	2.6e-04	1.1e+10	-2.4e+01	2.8e+01	693639	569480	55.03	44.93	498.31	49.79
11500	2.5e-04	1.1e+10	-8.9e+00	2.4e+01	673910	597322	54.77	48.20	521.32	53.85
11800	2.5e-04	1.2e+10	-3.9e+01	1.7e+01	757749	627850	62.92	52.02	581.98	55.67
12100	2.4e-04	1.2e+10	-2.9e+01	2.6e+01	748235	601448	63.64	51.45	640.96	57.59
12400	2.3e-04	1.2e+10	-4.3e+01	2.5e+01	777435	614047	68.12	54.42	653.76	60.37
12700	2.3e-04	1.3e+10	-3.3e+01	2.5e+01	775871	620193	70.07	56.35	686.85	63.40
13000	2.2e-04	1.3e+10	-3.2e+01	1.1e+01	771101	681834	73.02	65.22	703.04	68.91
13300	2.2e-04	1.3e+10	-2.7e+01	2.4e+01	766793	648685	75.93	64.69	727.64	71.98
13600	2.1e-04	1.4e+10	-3.5e+01	1.0e+01	814120	708483	82.37	72.15	752.62	75.16
13900	2.1e-04	1.4e+10	-5.5e+01	2.3e+01	863835	662119	89.35	68.57	824.58	76.07
14200	2.0e-04	1.4e+10	-4.4e+01	2.3e+01	858674	673736	90.12	71.18	887.64	78.25
14500	2.0e-04	1.5e+10	-4.3e+01	9.4e+00	872710	741010	94.29	80.77	869.02	84.41
14800	2.0e-04	1.5e+10	-3.0e+01	8.5e+00	836054	747797	91.59	82.12	874.94	84.96

4.3. LSQR Comparison: Algorithm 3.1 vs. Two-Sided κ-optimal Scaling in [25]. In Subsection 4.3, we compare the efficiency of our two-sided Algorithm 3.1 which reduces ω at every iteration, with the two-sided κ-optimal preconditioner presented in [25]. We consider matrices $A \in \mathcal{M}^n$ that are nonsingular and compare the performance of both preconditioners for minimizing $\|b - Ax\|$ with LSQR. The results presenting the comparison are presented in Tables 4.6 and 4.7. Table 4.6 considers matrices from the SuiteSparse Matrix Collection while Table 4.7 considers matrices using MATLAB's sprandn function. Both tables consider matrices where n does not exceed 300 since the two-sided κ-optimal algorithm of [25] could not handle larger matrices. In both tables, LSQR is run with a tolerance of 10^{-8} and a maximum iteration count of 5000.

Table 4.8 compares the performance of LSQR with no preconditioning with its performance using the

TABLE 4.6
LSQR Comparison on Small Suitesparse Matrices: Algorithm 3.1 vs [25]

Dim & Density, & $\kappa(A)$			Ratio of Condition Numbers		CPU Time for Prec		LSQR Iterations		Total CPU Time	
n	density	$\kappa(A)$	$\kappa(S)/\kappa(T)$	$\omega(S)/\omega(T)$	Algorithm 3.1	[25]	Algorithm 3.1	[25]	Algorithm 3.1	[25]
300	6.6e-03	5.2e+03	1.3e+00	1.0e+00	0.006	23.358	5	4	0.015	23.36
130	6.1e-02	6.1e+10	1.1e-01	8.2e-01	0.002	27.943	9	35	0.005	27.95
200	2.0e-02	2.4e+03	1.0e+00	1.0e+00	0.000	19.308	750	756	0.022	19.32
104	9.2e-02	5.5e+03	1.6e+00	9.7e-01	0.001	10.056	301	288	0.004	10.06
216	9.3e-02	3.3e+04	8.9e-01	4.6e-01	0.002	309.028	56	175	0.004	309.03
115	9.3e-02	3.7e+02	1.5e+00	6.7e-01	0.001	14.324	267	363	0.004	14.33
185	2.8e-02	1.8e+05	5.7e+00	7.5e-01	0.003	40.382	242	328	0.006	40.39
216	1.7e-02	1.0e+02	1.3e+00	9.0e-01	0.001	17.672	183	215	0.004	17.67
207	1.3e-02	1.4e+08	3.9e-01	3.4e-01	0.003	19.862	122	709	0.006	19.87
137	2.1e-02	1.8e+04	1.2e+00	7.9e-01	0.002	12.505	61	95	0.003	12.51
225	2.6e-02	7.1e+06	2.5e+00	8.1e-01	0.003	146.868	78	86	0.004	146.87
198	1.4e-01	3.0e+03	1.3e+00	5.8e-01	0.002	88.701	631	973	0.010	88.71
265	2.5e-02	1.4e+03	1.6e+00	5.0e-01	0.002	38.714	735	1379	0.010	38.73
100	4.0e-02	1.5e+04	1.0e+00	9.7e-01	0.000	8.944	114	132	0.002	8.95
105	8.0e-02	7.2e+02	1.5e+00	6.6e-01	0.000	14.751	163	227	0.002	14.75
135	3.6e-02	9.2e+05	1.0e+00	2.3e-01	0.000	17.484	164	850	0.002	17.49
120	6.0e-02	4.3e+08	3.1e-03	2.7e-02	0.001	24.967	77	845	0.003	24.97
100	7.1e-02	2.4e+12	5.8e-02	2.4e-01	0.001	28.979	19	108	0.003	28.98
136	2.6e-02	2.5e+05	1.9e+00	8.0e-01	0.001	19.051	132	175	0.003	19.05
100	4.0e-02	1.3e+04	2.3e+00	9.3e-01	0.000	10.800	339	385	0.003	10.80
300	3.5e-02	8.5e+05	2.2e+00	6.0e-01	0.002	581.259	1195	1983	0.016	581.28
132	2.4e-02	4.2e+11	2.1e+00	1.0e-01	0.000	22.553	93	1356	0.002	22.56

TABLE 4.7
LSQR Comparison on Small Random Matrices: Algorithm 3.1 vs [25]

Dim & Density, & $\kappa(A)$			Ratio of Condition Numbers		CPU Time for Prec		LSQR Iterations		Total CPU Time	
n	density	$\kappa(A)$	$\kappa(S)/\kappa(T)$	$\omega(S)/\omega(T)$	Algorithm 3.1	[25]	Algorithm 3.1	[25]	Algorithm 3.1	[25]
50	2.5e-01	1.0e+03	1.2e+00	9.4e-01	0.007	16.583	18	18	0.02	16.59
60	2.4e-01	1.0e+03	5.2e+00	5.1e-01	0.001	11.914	46	67	0.01	11.92
70	2.3e-01	1.1e+03	1.4e+00	8.3e-01	0.001	17.909	22	27	0.00	17.91
80	2.2e-01	1.1e+03	2.0e+00	2.6e-01	0.002	13.848	51	191	0.00	13.85
90	2.2e-01	1.2e+03	2.2e+00	5.0e-01	0.002	18.674	43	90	0.00	18.68
100	2.1e-01	1.2e+03	5.2e+00	4.9e-01	0.003	27.400	52	72	0.00	27.40
110	2.1e-01	1.3e+03	2.2e+00	6.4e-01	0.003	45.059	52	52	0.00	45.06
120	2.0e-01	1.4e+03	2.0e+00	7.0e-01	0.002	37.248	63	57	0.00	37.25
130	2.0e-01	1.5e+03	7.3e+00	4.0e-01	0.002	42.239	96	140	0.00	42.24
140	1.9e-01	1.6e+03	1.1e+01	5.4e-01	0.002	55.521	54	83	0.00	55.52
150	1.9e-01	1.7e+03	2.4e+00	8.0e-01	0.001	102.427	46	35	0.00	102.43
160	1.9e-01	1.8e+03	6.2e+00	5.3e-01	0.002	90.146	66	81	0.00	90.15
170	1.9e-01	1.9e+03	2.4e+01	5.1e-01	0.003	101.889	84	111	0.01	101.89
180	1.8e-01	2.1e+03	9.3e+00	4.3e-01	0.001	128.101	90	109	0.00	128.10
190	1.8e-01	2.2e+03	8.1e+00	3.4e-01	0.001	116.720	113	219	0.00	116.72
200	1.8e-01	2.5e+03	9.1e+00	4.7e-01	0.001	147.824	94	124	0.00	147.83
210	1.8e-01	2.7e+03	1.2e+01	5.2e-01	0.002	191.344	121	113	0.00	191.35
220	1.8e-01	3.1e+03	4.6e+00	5.3e-01	0.002	293.214	94	75	0.00	293.22
230	1.7e-01	3.5e+03	2.2e+00	3.8e-01	0.001	175.652	105	508	0.00	175.66
240	1.7e-01	4.0e+03	6.1e+00	4.5e-01	0.001	287.572	129	116	0.00	287.57
250	1.7e-01	4.8e+03	1.1e+01	5.4e-01	0.002	433.302	107	92	0.00	433.30
260	1.7e-01	5.9e+03	4.0e+00	5.1e-01	0.002	384.679	139	166	0.01	384.68
270	1.7e-01	7.8e+03	3.6e+00	6.3e-01	0.002	431.242	121	75	0.00	431.24
280	1.7e-01	1.1e+04	1.5e+01	3.9e-01	0.002	498.242	153	298	0.01	498.25
290	1.7e-01	2.0e+04	1.6e+01	4.5e-01	0.002	750.635	178	128	0.01	750.64
300	1.7e-01	1.0e+05	1.0e+01	2.8e-01	0.002	751.572	197	264	0.01	751.58

preconditioner from Algorithm 3.1 on large matrices from the SuiteSparse Matrix Collection with $n \geqslant 2000$. The algorithm in [25] is not considered in Table 4.8 since it could not handle these large problems. Since these problems are more difficult, LSQR is run with a tolerance of 10^{-6} and a maximum iteration count of 100000. In all tables $S := (\hat{C}^{1/2} A \hat{D}^{1/2})^T (\hat{C}^{1/2} A \hat{D}^{1/2})$ and $T := (D_1^{1/2} A D_2^{-1/2})^T (D_1^{1/2} A D_2^{-1/2})$ where $\hat{C}^{1/2}$ and $\hat{D}^{1/2}$ (resp. $D_1^{1/2}$ and $D_2^{-1/2}$) are the preconditioners found by Algorithm 3.1 (resp. [25]). Also, in all tables, the total CPU time is computed as the sum of the CPU time for preconditioning and the CPU time of LSQR. Finally, the list of SuiteSprase matrices considered in Tables 4.6 and 4.8 can be found in Appendix C.

We see in Tables 4.6 and 4.7 that Algorithm 3.1 computed a preconditioner in significantly less time than the two-sided κ-optimal preconditioner in [25]. Moreover, reduced ω more significantly than [25] although [25] reduced κ more significantly. Interestingly enough, Algorithm 3.1 produced a preconditioner that resulted in fewer (or equal) LSQR iterations than [25] on 20 of 22 (resp. 18 of 26) instances presented in Table 4.6 (resp. Table 4.7). In summary, the results presented in Tables 4.6 and 4.7 display that ω-optimal preconditioners are much cheaper to compute than κ-optimal preconditioners and that the ω-condition number is also better correlated with LSQR iterations than the κ-condition number for minimizing $\|b - Ax\|$.

Table 4.8 shows that Algorithm 3.1 still computes effective preconditioners in negligible CPU time even on larger matrices with dimensions exceeding 2000. Moreover, Algorithm 3.1 reduced ω very significantly on 21 of the 25 instances presented in Algorithm 3.1. Finally, Table 4.8 shows that Algorithm 3.1 led to a significant reduction in LSQR iterations. The total CPU time using preconditioning, i.e., Algorithm 3.1's CPU+ LSQR CPU, was smaller than the LSQR's CPU time without preconditioning on 22 of the 25 instances.

TABLE 4.8
LSQR Comparison on Large Suitesparse Matrices: Algorithm 3.1 vs No Preconditioning

Dim & Density, & $\omega(A)$			Ratio of Omega & Prec CPU Time		LSQR Iter		Total CPU Time	
n	density	$\omega(A)$	$\omega(S)/\omega(A)$	Algorithm 3.1 CPU	No Prec	Algorithm 3.1	No Prec	Algorithm 3.1
14214	1.3e-03	5.1e+01	3.2e-02	1.0e-01	100000	8963	3.63e+01	3.43e+00
4119	2.1e-03	4.1e+05	3.6e-06	1.7e-02	5461	946	4.98e-01	1.04e-01
7479	1.2e-03	2.8e+04	4.8e-05	3.0e-02	9393	1758	1.85e+00	3.81e-01
9271	1.4e-03	8.2e+06	1.7e-07	4.1e-02	1798	377	4.44e-01	1.37e-01
4562	6.3e-03	2.3e+02	7.4e-03	1.3e-02	6497	1070	1.11e+00	2.00e-01
3072	1.3e-02	1.9e+00	1.0e+00	9.8e-03	1358	1357	5.43e-01	6.06e-01
11341	7.6e-04	6.0e+01	5.0e-02	3.9e-02	26647	3050	7.55e+00	9.09e-01
2048	2.9e-03	2.8e+00	9.3e-01	1.6e-03	4730	4187	2.10e-01	1.93e-01
8192	7.3e-04	2.8e+00	9.4e-01	6.1e-03	19233	16963	3.15e+00	2.79e+00
14734	4.4e-04	2.3e+00	7.2e-01	4.2e-02	10631	6883	3.43e+00	2.26e+00
2283	9.2e-03	3.6e+01	5.6e-02	9.1e-03	17604	1305	1.15e+00	9.74e-02
2000	2.0e-03	2.5e+00	1.0e+00	7.9e-04	1578	1578	6.33e-02	6.54e-02
2000	5.9e-03	4.2e+00	5.9e-01	3.5e-03	9674	3842	4.76e-01	1.95e-01
5000	2.4e-03	4.2e+00	5.7e-01	6.9e-03	29362	7483	4.56e+00	1.15e+00
7000	1.7e-03	4.3e+00	5.7e-01	9.3e-03	50422	13747	9.93e+00	2.82e+00
7000	1.7e-03	4.2e+00	5.7e-01	9.4e-03	48882	11272	9.71e+00	2.23e+00
3175	8.4e-03	3.0e+01	7.0e-02	9.3e-03	16004	2157	2.06e+00	2.90e-01
5952	6.3e-04	3.7e+02	4.6e-03	1.8e-02	13984	5192	1.55e+00	5.96e-01
13694	3.9e-04	1.8e+02	7.1e-03	8.0e-02	100000	2551	3.08e+01	8.70e-01
5832	9.0e-03	1.7e+00	1.0e+00	1.0e-02	2397	2390	6.25e-01	6.36e-01
9801	9.1e-04	1.5e+00	1.0e+00	2.6e-03	3132	3036	7.93e-01	7.72e-01
4000	5.5e-04	1.3e+09	7.5e-10	7.3e-03	7194	5	5.05e-01	8.63e-03
5940	2.4e-03	6.6e+00	4.6e-01	3.6e-02	100000	44539	1.74e+01	7.78e+00
4326	3.3e-03	1.1e+05	1.7e-05	1.4e-02	34107	2486	4.85e+00	3.69e-01
9800	6.0e-04	2.6e+00	9.9e-01	6.4e-03	18382	18177	4.37e+00	4.29e+00

TABLE 4.9
*PCG tol. 1e−7; Medium, PC; A, κ-opt **VS** J, ω-opt of A*

Dim & Density		Ratios in conds (J, A)		J: ω-opt of A		Ratios A/J	
n	density	$\kappa(J)/\kappa(A)$	$\omega(J)/\omega(A)$	iters	cpu	iters	cpu
5000	9.40e-03	2.98e+00	7.368e-01	19.4	3.927e-03	25.2	13.9
10000	7.44e-03	3.31e+00	7.362e-01	18.8	6.698e-03	36.6	30.5
15000	6.02e-03	2.64e+00	7.365e-01	17.0	1.418e-02	41.6	39.0
20000	4.57e-03	3.56e+00	7.363e-01	20.6	2.861e-02	41.6	41.1
25000	3.47e-03	3.71e+00	7.370e-01	20.6	8.006e-02	30.9	29.1
30000	2.42e-03	3.52e+00	7.362e-01	20.2	4.327e-02	48.0	43.0
35000	1.56e-03	2.77e+00	7.371e-01	16.0	4.174e-02	36.5	36.1
40000	8.93e-04	2.89e+00	7.364e-01	18.0	2.535e-02	46.7	38.9
45000	4.15e-04	3.93e+00	7.370e-01	21.8	1.820e-02	28.5	27.0
50000	4.12e-04	4.12e+00	7.361e-01	24.0	2.679e-02	49.5	43.7

4.4. From κ-optimal M to "Improve PCG" using ω-Optimal Scaling. We now study the effect of applying ω-optimal diagonal scaling to a κ-optimally diagonally scaled matrix M, where $M \succ 0$.[10] We find a κ-optimally diagonally scaled matrix $M \succ 0$ using Theorem 2.7, i.e., the maximal and minimal eigenvectors of M, x_1 and x_n, satisfy (2.5) and are chosen using (2.6). We then choose the remaining eigenvalues to be evenly spread out in the open interval $\lambda_i \in (\lambda_n, \lambda_1), i = 2, \ldots, n-1$, with corresponding orthonormal eigenvectors x_i, $i = 2, \ldots n-1$, chosen in the orthogonal complement of span$\{x_1, x_n\}$, i.e., x_i are orthogonal to both x_1 and x_n. We use a (sparse) QR factorization to obtain the remaining $n-2$ orthonormal eigenvectors. Therefore, the density of M cannot be determined accurately in advance. The time to generate a random problem is typically very small and is hence negligible. For a more precise description of how M is constructed, see our code.

After constructing M, we then apply ω-optimal diagonal preconditioning to M to implicitly get J as described above, i.e., $J = D^{1/2}MD^{1/2}$ where $D = \text{Diag}(1./\,\text{diag}(M))$. We then compare the number of PCG iterations of solving $Mx = b$ and $J(D^{-1/2}x) = D^{1/2}b$. Note that only the vector b and the matrices A and D are needed as input for MATLAB's built-in PCG function. Each linear system that we consider in our experiments is solved by PCG using 5 different initial points so the iteration counts and runtimes reported for each experiment are averaged across these 5 runs. For the computational results showing the comparison between A and J, see Tables 4.9 and 4.10.

We now give several remarks about the results presented in Tables 4.9 and 4.10. Interestingly, on every instance tested applying the Jacobi or ω-optimal diagonal preconditioning to A led to significant improvement in PCG's performance for solving the linear system even though $\kappa(A) < \kappa(J)$. On average, across the 10 medium sized test instances ppresented in Table 4.9, PCG performed 38.5 times more number of iterations on the κ-optimal linear system than it did on ω-optimally scaled linear system. Moreover, PCG also on average

[10] For the large problems in Table 4.10 we used the so-called fastlinux server, cpu155.math.private, a Dell PowerEdge R650 with two Intel Xeon Gold 6334 8-core 3.6 GHz (Ice Lake) and 256 GB.

TABLE 4.10
PCG tol. 1e−7; Large, Linux; A, κ-opt **VS** *J, ω-opt of A*

Dim & Density		Ratios in conds (J, A)		J: ω-opt of A		Ratios A/J	
n	density	$\kappa(J)/\kappa(A)$	$\omega(J)/\omega(A)$	iters	cpu	iters	cpu
60000	9.14e-03	4.27e+00	7.366e-01	18.2	8.894e-01	42.0	36.5
65000	7.78e-03	4.57e+00	7.362e-01	21.0	9.985e-01	52.9	47.9
70000	6.54e-03	3.93e+00	7.365e-01	17.0	8.110e-01	48.3	42.9
75000	5.44e-03	4.18e+00	7.363e-01	18.0	8.253e-01	56.4	49.2
80000	4.36e-03	4.08e+00	7.369e-01	17.0	7.202e-01	38.3	33.8
85000	3.47e-03	3.91e+00	7.362e-01	18.0	7.062e-01	60.9	54.0
90000	2.63e-03	3.92e+00	7.371e-01	17.6	6.050e-01	34.2	29.8
95000	1.91e-03	4.12e+00	7.364e-01	18.0	5.092e-01	48.0	41.6
100000	1.31e-03	4.13e+00	7.369e-01	18.2	4.029e-01	34.4	29.8
105000	8.09e-04	4.41e+00	7.361e-01	22.0	3.342e-01	58.8	50.3
110000	4.29e-04	3.58e+00	7.363e-01	17.0	1.709e-01	55.9	45.2
115000	1.74e-04	3.97e+00	7.369e-01	20.6	1.134e-01	31.6	23.7
120000	3.32e-05	3.14e+00	7.362e-01	20.2	3.648e-02	55.0	26.0

took 34.2 times longer (in terms of CPU time) on the κ-optimal linear system than it did on the ω-optimally scaled linear system.

Jacobi preconditioning led to even more significant improvements on the larger test instances presented in Table 4.10. On average, across the 13 large test instances in Table 4.10, PCG performed 47.4 times more number of iterations on the κ-optimal linear system than it did on ω-optimally scaled linear system. Moreover, PCG also on average took 39.3 times longer (in terms of CPU time) on the κ-optimal linear system than it did on the ω-optimally scaled linear system.

5. Conclusion. In this paper, we studied optimal diagonal preconditioning through the lens of two condition numbers: the classical κ and the averaging-based ω-condition number. On the theoretical side, we introduced an affine-based pseudoconvex reformulation of the κ-optimal preconditioning problem, yielding simple optimality conditions and enabling efficient optimization over an n-dimensional vector. Moreover, since the κ-condition number is positively homogeneous of degree zero, the associated minimization problem is Hadamard ill-posed. To address this, we introduce a reformulation that removes this homogeneity, leading to improved computational stability in practice. Building on this formulation, we develop a highly efficient subgradient method with convergence guarantees that significantly outperforms existing SDP-based approaches in both scalability and accuracy. For example, Table 4.1 shows that Algorithm 2.2 (resp. Algorithm B.1) is, on average, 77.48 (resp. 356.22) times faster than [25] when minimizing κ on relatively small SuiteSparse matrices. Our methods also substantially outperform [12] in terms of runtime. In addition to these speedups, our subgradient methods consistently achieve significantly greater reductions in κ than both [25] and [12], often reducing it by more than half. Finally, our algorithms scale to large problem instances with hundreds of thousands of variables, solving them within minutes, whereas existing methods [12, 25] struggle to handle such instances within a reasonable time.

In parallel, we provided explicit and unified characterizations of ω-optimal diagonal and block-diagonal preconditioners, showing that many classical schemes such as Jacobi scaling, row/column normalization, and matrix balancing arise naturally as ω-optimal solutions or stationary points of the ω-optimal preconditioning problem. These results offer a new perspective on widely used preconditioning techniques and, to the best of our knowledge, constitute the first comprehensive comparison of κ- and ω-based optimality conditions.

Our numerical results further reveal a clear and practically important message: while κ-optimal preconditioners reduce the worst-case condition number more aggressively, ω-optimal preconditioners are substantially cheaper to compute and more strongly correlated with the performance of iterative methods such as PCG and LSQR. Moreover, applying the ω-optimal diagonal scaling to linear systems that are already κ-optimally preconditioned leads to further significant improvements in PCG's performance.

Overall, our findings suggest that ω-based preconditioning provides a computationally efficient and practically superior alternative to κ-based approaches for large-scale problems, and highlight the importance of moving beyond worst-case conditioning when designing preconditioners.

Acknowledgments. The authors acknowledge the use of AI tools in revising the grammar and writing of this paper.

Appendix A. Assumptions and Technical Results from [19]. Let $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{-\infty, \infty\}$ be an extended real-valued function with *domain* $\bar{D}$. The following minimization problem is considered in [19]:

$$f_* := \inf \{f(x) : x \in X\}, \tag{A.1}$$

with the following assumptions on f and X:

A1 : $\operatorname{int} \bar{D}$ is nonempty and convex.
A2 : $\overline{\lim}_{t \downarrow 0} f(x + t(y - x)) \leqslant f(x)$, $\forall x \in \bar{D}, y \in \operatorname{int} \bar{D}$.
A3 : f is upper semicontinuous (usc) on $\operatorname{int} \bar{D}$, i.e., $f(x) = \lim_{\epsilon \downarrow 0} \sup_{B(x,\epsilon)} f$, $\quad \forall x \in \operatorname{int} \bar{D}$ where $B(x, \epsilon) := \{y : \|y - x\| \leqslant \epsilon\}$.
A4 : f is quasiconvex on $\operatorname{int} \bar{D}$, i.e., the set $\{x \in \operatorname{int} \bar{D} : f(x) \leqslant \alpha\}$ is convex $\forall \alpha \in \mathbb{R}$.
A5 : The constraint set $X \subset \mathbb{R}^n$ is closed convex, and $X \cap \operatorname{int} \bar{D} \neq \varnothing$.

The following lemma from [19] discusses that fractional programs with certain structures are quasiconvex and also provides one characterization of their quasisubgradients.

LEMMA A.1 ([19, Lemma 4]). *Suppose $f(x) = a(x)/b(x)$ for all $x \in \operatorname{int} \bar{D}$, where a is a convex function, b is finite and positive on $\operatorname{int} \bar{D}$, $\operatorname{int} \bar{D}$ is convex, and one of the following conditions holds:*

(a) b is affine;
(b) a is nonnegative on $\operatorname{int} \bar{D}$ and b is concave;
(c) a is nonpositive on $\operatorname{int} \bar{D}$ and b is convex.

Then f is quasiconvex on $\operatorname{int} \bar{D}$ and for each $x \in \operatorname{int} \bar{D}$, if $\alpha := f(x)$ is finite then $a - \alpha b$ is convex and $\partial[a - \alpha b](x) \subset \bar{\partial}^\circ f(x)$.

The following lemma can be found in [19] and is useful for showing that assumptions (A1)-(A4) hold for our set-up in Equation (2.8) since $\kappa(\mathring{\mathcal{D}}(d))$ is the ratio of a convex and concave function that is positive on Ω.

LEMMA A.2 ([19, Remark 2]). *Suppose a and $\tilde{b} := -b$ are proper convex functions on $\operatorname{int} \bar{D}^a \cap \operatorname{int} \bar{D}^b$, a is nonnegative on the (open and convex set) $C := \left\{y \in \operatorname{int} \bar{D}^a \cap \operatorname{int} \bar{D}^b : b(y) > 0\right\} \neq \varnothing$, and*

$$f(x) := \begin{cases} a(x)/b(x), & \text{if } x \in C, \\ \sup_{y \in C} \overline{\lim}_{t \downarrow 0} f(x + t(y - x)), & \text{if } x \in \operatorname{bd} C, \\ \infty, & \text{if } x \notin \operatorname{cl} C \end{cases}$$

*where $\operatorname{cl} C$ denotes the closure, $\operatorname{bd} C$ denotes the boundary of C, and $\bar{D}^a$ and $\bar{D}^b$ denote the domains of a and b, respectively. Then assumptions **A1-A4** hold with $\operatorname{int} \bar{D} = C$.*

Appendix B. An Efficient Line-Search Subgradient Method. This section presents an efficient line-search subgradient method for minimizing $\kappa(v) := \kappa(\mathring{\mathcal{V}}(v))$ where $\mathring{\mathcal{V}}(v)$ is as in Equation (2.15).

Let $w = e + Vv \in \mathbb{R}^n_{++}$ correspond to the current iterate v. Also, let σ be a small scalar between 0 and 1 and let $\Delta w = V \Delta v$ where $\Delta v = -\nabla \kappa(v)$ where $\kappa(v)$ is as in Equation (2.16). From Lemma 2.3, we can use a ratio test and guarantee positive definiteness from $w + tV\Delta v = w + t\Delta w > \sigma w$, or equivalently $t\Delta w > (\sigma - 1)w$. Therefore, we conclude that the maximum step with safeguarding, so as not to get too close to the boundary, is

$$t_{\max}(\sigma) \cong t_{\max} := \min \left\{ \frac{(\sigma - 1)w_i}{\Delta w_i} : \Delta w_i < 0 \right\} > 0. \tag{B.1}$$

Recall the gradient $\nabla \kappa(v) = \nabla(\kappa \circ \mathring{\mathcal{V}}(v))$ in Lemma 2.14. Our line search in Algorithm B.1 maintains:

LineSearch B.1 (for Algorithm B.1). Backtrack and maintain: (i) $t \leqslant t_{\max}$ from (B.1); (ii) monotonic nonincrease of the objective $\kappa(v + t\Delta v)$; and nonpositivity of the directional derivative $\nabla \kappa(v + t\Delta v)^T \Delta v \leqslant 0$.

Algorithm B.1 is presented below.

Algorithm B.1 An Efficient Line-Search Subgradient Method for Minimizing $\kappa(v)$.

Inputs: A symmetric positive definite matrix $A \succ 0$, a max iteration count mainmaxiter, stopping tolerances maintoler> 0 and linesrchtoler> 0, and a small scalar $0 < \sigma \ll 1$.

set $k \leftarrow 0$, $v_1 = 0 \in \mathbb{R}^{n-1}$, $w_1 = e_n \in \mathbb{R}^n$, and $t_k \leftarrow \infty$;
compute a min eigenpair (λ_n^1, x_n^1) and max eigenpair (λ_1^1, x_1^1) of A;
compute $\kappa(v_1)$ and $\nabla\kappa(v_1)$ according to Equation (2.16) and Equation (2.19), respectively;
set relnormg$= \|\nabla\kappa(v_1)\|^2/(1+\kappa(v_1)^2)$;
while relnormg $>$ maintoler & $t_k >$ linesrchtoler & $k \leqslant$ mainmaxiter **do**
  set $k \leftarrow k+1$;
  set $\Delta v_k \leftarrow -\nabla\kappa(v_k)$;
  use (B.1) with $\Delta w := V\Delta v_k$ and $w := w_k$ to evaluate $t_{\max}(\sigma)$;
  perform LineSearch B.1 to find t_k;
  set $v_{k+1} = v_k + t_k\Delta v_k$ and $w_{k+1} = e_n + Vv_{k+1}$;
  compute min eigenpair $(\lambda_n^{k+1}, x_n^{k+1})$ and max eigenpair $(\lambda_1^{k+1}, x_1^{k+1})$ of $A\,\mathrm{Diag}(w_{k+1})$;
  compute $\kappa(v_{k+1})$ and $\nabla\kappa(v_{k+1})$ according to Equation (2.16) and Equation (2.19), respectively;
  update relnormg$= \|\nabla\kappa(v_{k+1})\|^2/(1+\kappa(v_{k+1})^2)$;
end while(main outer loop)

Output: $\hat{D} := \mathrm{Diag}(w_{k+1})$.

In practice, we take linesearchtol to be 10^{-7} and maintoler to be 10^{-4}.

Appendix C. List of SuiteSparse Matrices Used in Experiments. Table 4.1 considers the following matrices:

"bcsstk08.mat", "bcsstk13.mat", "bcsstk21.mat", "bcsstk23.mat", "bcsstk24.mat"
"bcsstk26.mat", "bcsstk28.mat", "bcsstk34.mat", "494_bus.mat", "662_bus.mat"
"nasa1824.mat", "nasa2146.mat", "nasa2910.mat", "nos1.mat", "nos2.mat", "nos4.mat"
"nos5.mat", "nos7.mat"

Table 4.3 considers the following matrices:

"Pres_Poisson.mat", "bcsstk25.mat", "bcsstm25.mat", "gyro_m.mat"
"gyro.mat", "bcsstk36.mat", "wathen100.mat", "wathen120.mat", "minsurfo.mat"
"gridgena.mat", "G2_circuit.mat"

Table 4.6 considers the following matrices:

"08blocks.mat", "arc130.mat", "bwm200.mat", "ck104.mat", "ex1.mat"
"football.mat", "gre_185.mat", "gre_216a.mat", "impcol_a.mat", "impcol_c.mat"
"impcol_e.mat", "jazz.mat", "lshp_265.mat", "olm100.mat", "polbooks.mat"
"rajat11.mat", "robot.mat", "rotor1.mat", "rw136.mat", "tub100.mat"
"utm300.mat", "west013.mat"

Table 4.8 considers the following matrices:

"airfold_2d.mat", "c-24.mat", "c-36.mat", "c-39.mat", "cavity21.mat"
"conf5_-4x4-18.mat", "coupled.mat", "delaunay_n11.mat", "delaunay_n13.mat", "epb1.mat"
"ex24.mat", "G34.mat", "G36.mat", "G59.mat", "G63.mat"
"G64.mat", "garon1.mat", "Hamrle2.mat", "jan99jac040sc.mat", "Na5.mat"
"t2d_q9.mat", "tols4000.mat", "utm5940.mat", "viscoplastic1.mat"
"whitaker3.mat"

[1] M. Benzi, *Preconditioning techniques for large linear systems: a survey*, J. Comput. Phys., 182 (2002), pp. 418–477, https://doi.org/10.1006/jcph.2002.7176, https://doi-org.proxy.lib.uwaterloo.ca/10.1006/jcph.2002.7176.

[2] A. Bock and M. Andersen, *Connecting Kaporin's condition number and the Bregman log determinant divergence*, tech. report, 2025, https://arxiv.org/abs/2503.22286, https://arxiv.org/abs/2503.22286.

[3] R. Byrd and R. Nocedal, *A tool for the analysis of quasi-Newton methods with application to unconstrained minimization*, SIAM J. Numer. Anal., 26 (1989), pp. 727–739.

[4] R. Byrd, R. Nocedal, and Y.Yuan, *Global convergence of a class of quasi-Newton methods on convex problems*, SIAM J. Numer. Anal., 24 (1987), pp. 1171–1191.

[5] G. Ciaramella and M. J. Gander, *Iterative methods and preconditioners for systems of linear equations*, vol. 19 of Fundamentals of Algorithms, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, [2022] ©2022, https://doi.org/10.1137/1.9781611976908, https://doi-org.proxy.lib.uwaterloo.ca/10.1137/1.9781611976908.

[6] T. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Transactions on Mathematical Software (TOMS), 38 (2011), pp. 1–25.

[7] J. E. Dennis, Jr. and H. Wolkowicz, *Sizing and least-change secant methods*, SIAM J. Numer. Anal., 30 (1993), pp. 1291–1314, https://doi.org/10.1137/0730067, https://doi-org.proxy.lib.uwaterloo.ca/10.1137/0730067.

[8] X. Doan and H. Wolkowicz, *Numerical computations and the ω-condition number*, Tech. Report CORR 2011-03, University of Waterloo, Waterloo, Ontario, 2011. www.math.uwaterloo.ca/~hwolkowi/henry/reports/ONE.pdf.

[9] X. V. Doan, S. Kruk, and H. Wolkowicz, *A robust algorithm for semidefinite programming*, Optim. Methods Softw., 27 (2012), pp. 667–693, https://doi.org/10.1080/10556788.2011.610456, https://doi-org.proxy.lib.uwaterloo.ca/10.1080/10556788.2011.610456.

[10] A. Franceschini, M. Ferronato, C. Janna, V. A. Paludetto Magri, et al., *Recent advancements in preconditioning techniques for large size linear systems suited for high performance computing*, Dolomites Research Notes on Approximation, 11 (2018), pp. 11–22.

[11] W. Gao, Y.-C. Chu, Y. Ye, and M. Udell, *Gradient methods with online scaling*, tech. report, 2024, https://arxiv.org/abs/2411.01803, https://arxiv.org/abs/2411.01803. arXiv, 2411.01803.

[12] W. Gao, Z. Qu, M. Udell, and Y. Ye, *Scalable approximate optimal diagonal preconditioning*, tech. report, 2023, https://arxiv.org/abs/2312.15594. arXiv, 2312.15594.

[13] J. A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[14] A. Greenbaum, *Iterative methods for solving linear systems*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[15] Y. Hu, J. Li, and C. K. W. Yu, *Convergence rates of subgradient methods for quasi-convex optimization problems*, Comput. Optim. Appl., 77 (2020), pp. 183–212, https://doi.org/10.1007/s10589-020-00194-y, https://doi-org.proxy.lib.uwaterloo.ca/10.1007/s10589-020-00194-y.

[16] C. G. J. Jacobi, *Ueber eine neue auflösungsart der bei der methode der kleinsten quadrate vorkommenden lineären gleichungen*, Astronomische Nachrichten, 22 (1845), p. 297–306.

[17] W. L. Jung, D. Torregrosa-Belén, and H. Wolkowicz, *The ω-condition number: applications to preconditioning and low rank generalized Jacobian updating*, Comput. Optim. Appl., 91 (2025), pp. 235–282, https://doi.org/10.1007/s10589-025-00669-w, https://doi-org.proxy.lib.uwaterloo.ca/10.1007/s10589-025-00669-w.

[18] I. Kaporin, *New convergence results and preconditioning strategies for the conjugate gradient method*, Numer. Linear Algebra Appl., 1 (1994), pp. 179–210, https://doi.org/10.1002/nla.1680010208, https://doi.org/10.1002/nla.1680010208.

[19] K. Kiwiel, *Convergence and efficiency of subgradient methods for quasiconvex minimization*, Mathematical programming, 90 (2001), pp. 1–25.

[20] P. A. Knight, *The Sinkhorn–Knopp algorithm: convergence and applications*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 261–275.

[21] P. A. Knight, D. Ruiz, and B. Uçar, *A symmetry preserving algorithm for matrix scaling*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 931–955, https://doi.org/10.1137/110825753, https://doi-org.proxy.lib.uwaterloo.ca/10.1137/110825753.

[22] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM journal on scientific computing, 23 (2001), pp. 517–541.

[23] J. Lee, *Smooth manifolds*, in Introduction to smooth manifolds, Springer, 2003, pp. 1–29.

[24] MOSEK ApS, *MOSEK Optimization Suite*, 2025, https://www.mosek.com. Version 11.1 (or the version used).

[25] Z. Qu, W. Gao, O. Hinder, Y. Ye, and Z. Zhou, *Optimal diagonal preconditioning*, Operations Research, 73 (2024), pp. 1479–1495, https://doi.org/10.1287/opre.2022.0592, https://doi.org/10.1287/opre.2022.0592, https://arxiv.org/abs/https://doi.org/10.1287/opre.2022.0592.

[26] Y. Saad, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, second ed., 2003, https://doi.org/10.1137/1.9780898718003, https://doi-org.proxy.lib.uwaterloo.ca/10.1137/1.9780898718003.

[27] Y. Saad, *Iterative methods for linear systems of equations: a brief historical journey*, 754 ([2020] ©2020), pp. 197–215, https://doi.org/10.1090/conm/754/15141, https://doi-org.proxy.lib.uwaterloo.ca/10.1090/conm/754/15141.

[28] M. Scetbon, C. Ma, W. Gong, and E. Meeds, *Gradient multi-normalization for stateless and scalable llm training*, arXiv preprint arXiv:2502.06742, (2025).

[29] G. W. Soules, *The rate of convergence of Sinkhorn balancing*, in Proceedings of the First Conference of the International Linear Algebra Society (Provo, UT, 1989), vol. 150, 1991, pp. 3–40, https://doi.org/10.1016/0024-3795(91)90157-R, https://doi-org.proxy.lib.uwaterloo.ca/10.1016/0024-3795(91)90157-R.

[30] L. Tunçel and H. Wolkowicz, *Strengthened existence and uniqueness conditions for search directions in semidefinite programming*, Linear Algebra Appl., 400 (2005), pp. 31–60, https://doi.org/10.1016/j.laa.2004.12.010, https://doi-org.proxy.lib.uwaterloo.ca/10.1016/j.laa.2004.12.010.

[31] N. van der Aa, H. ter Morsche, and R. Mattheij, *Computation of eigenvalue and eigenvector derivatives for a general complex-valued eigensystem*, Electron. J. Linear Algebra, 16 (2007), pp. 300–314, https://doi.org/10.13001/1081-3810.1203, https://doi.org/10.13001/1081-3810.1203.
[32] A. Wathen, *Preconditioning*, Acta Numer., 24 (2015), pp. 329–376, https://doi.org/10.1017/S0962492915000021, https://doi.org/10.1017/S0962492915000021.
[33] H. Wolkowicz and Q. Zhao, *An all-inclusive efficient region of updates for least change secant methods*, SIAM J. Optim., 5 (1995), pp. 172–191, https://doi.org/10.1137/0805009, https://doi-org.proxy.lib.uwaterloo.ca/10.1137/0805009.
[34] W. E. Young and R. H. Trent, *Geometric mean approximations of individual security and portfolio performance*, The Journal of Financial and Quantitative Analysis, 4 (1969), pp. 179–199, http://www.jstor.org/stable/2329839 (accessed 2026-03-03).
[35] Q. Zhao, *Measures for least change secant methods*, master's thesis, University of Waterloo, 1993.