Noname manuscript No.
(will be inserted by the editor)

A Feasible Method for Optimization with Orthogonality Constraints

Zaiwen Wen · Wotao Yin

Received: date / Accepted: date

Abstract Minimization with orthogonality constraints (e.g., $X^\top X = I$) and/or spherical constraints (e.g., $\|x\|_2 = 1$) has wide applications in polynomial optimization, combinatorial optimization, eigenvalue problems, sparse PCA, p-harmonic flows, 1-bit compressive sensing, matrix rank minimization, etc. These problems are difficult because the constraints are not only non-convex but numerically expensive to preserve during iterations. To deal with these difficulties, we apply the Cayley transform — a Crank-Nicolson-like update scheme — to preserve the constraints and based on it, develop curvilinear search algorithms with lower flops compared to those based on projections and geodesics. The efficiency of the proposed algorithms is demonstrated on a variety of test problems. In particular, for the maxcut problem, it exactly solves a decomposition formulation for the SDP relaxation. For polynomial optimization, nearest correlation matrix estimation and extreme eigenvalue problems, the proposed algorithms run very fast and return solutions no worse than those from their state-of-the-art algorithms. For the quadratic assignment problem, a gap 0.842% to the best known solution on the largest problem "tai256c" in QAPLIB can be reached in 5 minutes on a typical laptop.

Keywords Orthogonality constraint · spherical constraint · Stiefel manifold · Cayley transformation · curvilinear search · polynomial optimization · maxcut SDP · nearest correlation matrix · eigenvalue and eigenvector · invariant subspace · quadratic assignment problem

Mathematics Subject Classification (2010) 49Q99 · 65K05 · 90C22 · 90C26 · 90C27 · 90C30

The work of Z. Wen was supported in part by NSF DMS-0439872 through UCLA IPAM. The work of W. Yin was supported in part by NSF grants DMS-07-48839 and ECCS-10-28790, and ONR Grant N00014-08-1-1101.

Zaiwen Wen
Department of Mathematics and Institute of Natural Sciences, Shanghai Jiaotong University, China
E-mail: zw2109@sjtu.edu.cn

Wotao Yin
Department of Computational and Applied Mathematics, Rice University, Texas, 77005, U.S.A.
E-mail: wotao.yin@rice.edu

1 Introduction

Matrix orthogonality constraints play an important role in many applications of science and engineering. In this paper, we consider optimization with orthogonality constraints:

$$\min_{X\in\mathbb{R}^{n\times p}} \quad \mathcal{F}(X),\ \text{s.t. } X^\top X = I, \tag{1}$$

where I is the identity matrix and $\mathcal{F}(X) : \mathbb{R}^{n\times p} \to \mathbb{R}$ is a differentiable function. The feasible set $\mathcal{M}_n^p := \{X \in \mathbb{R}^{n\times p} : X^\top X = I\}$ is often referred to as the Stiefel manifold, which reduces to the unit-sphere manifold $\mathrm{Sp}^{n-1} := \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$ when $p = 1$. We also consider the generalized orthogonality constraints $X^*MX = K$, where $X \in \mathbb{C}^{n\times p}$, $M \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix, and $K \in \mathbb{C}^{p\times p}$ is a nonsingular Hermitian matrix. The corresponding minimization problem becomes

$$\min_{X\in\mathbb{C}^{n\times p}} \quad \mathcal{F}(X),\ \text{s.t. } X^*MX = K. \tag{2}$$

When there are more than one such constraint, the problem is written as

$$\min_{X_1\in\mathbb{C}^{n_1\times p_1},\cdots,X_q\in\mathbb{C}^{n_q\times p_q}} \mathcal{F}(X_1,\cdots,X_q),\ \text{s.t. } X_1^*M_1X_1 = K_1,\ldots,X_q^*M_qX_q = K_q, \tag{3}$$

where M_i and K_i, $i = 1,\ldots,q$, are given positive definite and nonsingular symmetric matrices.

It is generally difficult to solve problems (1), (2) and (3) since the orthogonality constraints can lead to many local minimizers and, in particular, several of these problems in special forms are NP-hard. There is no guarantee for obtaining the global minimizer except for a few simple cases (e.g., finding the extreme eigenvalues). Even generating a sequence of feasible points is not easy since preserving the orthogonality constraints can be numerically expensive. Most existing constraint–preserving algorithms either use matrix re-orthogonalization or generate points along geodesics of $\mathcal{M}_n^p$. The former requires matrix factorizations such as the singular value decompositions (SVDs), and the latter must compute matrix exponentials or solve partial differential equations (PDEs). In this paper, we develop optimization algorithms based on the Cayley transform for preserving the constraints. For simplicity of exposition but no fundamental reasons, we present our algorithms and analysis for the simpler problem (1). The results apply to the more general problems (2) and (3) after trivial changes.

1.1 Constraint-Preserving Update

Given a feasible point X and the gradient $G := \mathcal{D}\mathcal{F}(X) = \left(\frac{\partial \mathcal{F}(X)}{\partial X_{i,j}}\right)$, we define a skew-symmetric matrix A as either

$$A := GX^\top - XG^\top \quad \text{or} \tag{4}$$

$$A := (P_XG)X^\top - X(P_XG)^\top, \quad \text{where } P_X := (I - \frac{1}{2}XX^\top). \tag{5}$$

The new trial point is determined by the Crank-Nicolson-like scheme

$$Y(\tau) = X - \frac{\tau}{2}A\left(X + Y(\tau)\right), \tag{6}$$

where $Y(\tau)$ is given in the closed form:

$$Y(\tau) = QX \quad \text{and} \quad Q := (I + \frac{\tau}{2}A)^{-1}(I - \frac{\tau}{2}A). \tag{7}$$

Known as the Cayley transformation, Q leads to several nice properties: the curve $Y(\tau)$ is smooth in τ, $Y(0) = X$, and most importantly $(Y(\tau))^\top Y(\tau) = X^\top X$ for all $\tau \in \mathbb{R}$. Moreover, $\frac{\mathrm{d}}{\mathrm{d}\tau}Y(0)$ equals the projection of $(-G)$ into the tangent space of $\mathcal{M}_n^p$ at X (the two different definitions of A in (4) and (5) correspond to two different metrics of the tangent space); hence $\{Y(\tau)\}_{\tau \geq 0}$ is a descent path. Similar to line search along a straight line, curvilinear search can be applied to finding a proper step size τ and guaranteeing the iterations to converge to a stationary point.

Formulation (7) requires inverting $(I + \frac{\tau}{2}A)$ in order to preserve the orthogonality constraints. It is often numerically cheaper than computing SVDs or geodesics. When $p = 1$, the inverse $(I + \frac{\tau}{2}A)^{-1}$ (and thus $Y(\tau)$) is given in a closed form with a few vector-vector products. When p is much smaller than $n/2$ — which is the case in many applications — A has rank $2p$; hence it follows from the Sherman-Morrison-Woodbury (SMW) theorem that one only needs to invert a smaller $2p \times 2p$ matrix. The total flops for computing $Y(\tau)$ is $4np^2 + O(p^3)$, and updating $Y(\tau)$ for a different τ needs $2np^2 + O(p^3)$ flops. When p is greater than or approximately equal to $n/2$, we can either compute $(I + \frac{\tau}{2}A)^{-1}$ directly or apply the SMW theorem recursively. Furthermore, since orthogonality is preserved by any skew-symmetric matrix A, there exist many different ways to generate low-rank matrices A and descent curves $Y(\tau)$ that are easy to compute.

Preliminary numerical experiments on a wide collection of problems show that the proposed algorithms have very promising performance. While global minimization cannot be guaranteed, our algorithms run very fast and often return solutions and objective values no worse than those obtained from specialized solvers for a variety of problems. Global optimality can be proved for a low-rank decomposition formulation for the maxcut SDP relaxation. It is worth mentioning that the proposed algorithm returns very good approximate solutions rapidly for the quadratic assignment problem (QAP), which is rewritten with constraints $X^\top X = I$ and $X \geq 0$. It took merely 5 minutes on a laptop to reach a gap of 0.842% to the best known solution for the largest problem "tai256c" in QAPLIB.

1.2 Relationships to Existing Work

1.2.1 Cayley transformation

The update formula (7) is widely known as the Cayley transformation, which has been long used in matrix computations such as inverse eigenvalue problems [21]. It is also dubbed as the Crank-Nicholson scheme for solving the heat equation and similar PDEs. Despite its long existence, the formula has not been widely applied for solving (1) or developed into practical algorithms and codes. Also, it has not been considered to take general update directions other than those based on the gradients such as (4) and (5).

We developed the update from our previous work on p-harmonic flow (see next subsection) by extending its update formula to handle matrices. We soon became aware that the update is the Cayley transform. Later, an

anonymous referee brought to our attention the work [39] on learning algorithms for neural networks, which also uses the same update. Compared to existing work, the contributions of this paper include further developing the update with monotone and non-monotone curvilinear search, efficient implementations especially for the case $p \ll n$, some extensions and generalizations, as well as numerical experiments on a wide collection of problems with comparisons to their state-of-the-art algorithms.

1.2.2 p-Harmonic Flow

The formulas (4)-(7) can be obtained by extending the authors' previous work [23] for finding a p-harmonic flow $U(x)$, which is the minimizer of $E(U) := \int_\Omega \|\nabla U(x)\|_2^p \mathrm{d}x$ subject to the point-wise unit spherical constraints $\|U(x)\|_2 = 1$ for $x \in \Omega$. For $U(x) \in \mathbb{R}^3$, [23] introduces the formula

$$U_{k+1} = U_k - \tau\,(U_k \times \mathcal{D}E_k) \times \frac{U_{k+1} + U_k}{2}, \tag{8}$$

where $\times$ is the cross-product operator and $\mathcal{D}E_k := \mathcal{D}\cdot(\|\mathcal{D}U\|^{p-2}\mathcal{D}U)$ (where $\mathcal{D}\cdot$ stands for divergence) is the Fréchet derivative of $E(U)$ at U_k. It is shown in [23] that $\|U_{k+1}(x)\|_2 = \|U_k(x)\|_2$, $\forall x \in \Omega$. Since $(x \times g) \times y = (gx^\top - xg^\top)y$ for vectors $x, g, y \in \mathbb{R}^3$, we can rewrite the last term in (8) as

$$\tau(U_k \times \mathcal{D}E_k) \times \frac{U_{k+1} + U_k}{2} = \frac{\tau}{2}((\mathcal{D}E_k)(U_k)^\top - U_k(\mathcal{D}E_k)^\top)(U_{k+1} + U_k),$$

which looks nothing but the last term in (6) with A given by (4).

1.2.3 Optimization on Manifolds

The term "optimization on manifolds" is applied to algorithms that preserve manifold constraints during iterations. Generally speaking, preserving constraints has advantages in many situations, for example, when the cost functions are undefined or of little use outside the feasible region and when the algorithm is terminated prior to convergence yet a feasible solution is required. Theories and methods for optimization on manifolds date back to the 1970s, and algorithms for specific problems, e.g., various eigenvalue problems, appeared even earlier. Most of the general-purpose algorithms for (1), however, did not appear until the 1990s [2].

While it is trivial to generate trial points in $\mathbb{R}^n$ along straight search lines, it is not as easy to do so in the curved manifold $\mathcal{M}_n^p$. A natural choice is the geodesic, which is the analog of straight line: it has the shortest length between two points in the manifold. Unfortunately, points on a geodesic of $\mathcal{M}_n^p$ are difficult to compute. As a relaxation of geodesic, retraction [2] smoothly maps a tangent vector to the manifold. Our $Y(\tau)$ is a retraction; see page 59 of [2] for the definition. Specifically, as discussed in subsection 4.1, given any tangent direction D of the manifold at the current point X, we obtain a curve $Y(\tau)$ on the manifold via (15) in which we set $W = P_X D X^\top - X D^\top P_X$ (where P_X is defined in (5)), and $Y(\tau)$ obeys $Y'(0) = -D$. The two specific versions of $Y(\tau)$ corresponding to (4) and (5) can be recovered in this way by setting D as the projections of $\nabla_X \mathcal{F}$ to the tangent space at X under the canonical metric and the Euclidean metric, respectively; see subsection 4.1. Comparing to other retraction-based methods [1–4, 48], the computational cost of ours can be cheaper under certain cases. It is worth noting that while $Y(\tau)$ given by (4) or (5) is generally not a geodesic of

$\mathcal{M}_n^p$ for $p > 2$, in the vector case $p = 1$ it is a geodesic of Sp^{n-1} — the geodesic rooted at X along the tangent direction $(-AX)$. Also note that in the latter case, τ is not the geodesic length.

There are various optimization methods in Riemannian manifolds that apply to (1)–(3), and most of them rely on geodesics or projections. The rather simple methods are based on the steepest descent gradient approach; see, for example, [2,29,53] and references therein. The conjugate gradient method and Quasi-Newton method have been extended to Riemannian manifolds in [18] and [9,43], respectively. Newton's methods, such as those in [2,18,41,50,51,53], use the second-order information of the objective function to achieve superlinear convergence. Riemannian and implicit Riemannian trust-region methods are proposed in [1,4]. Since the second-order methods may require some additional computation depending on the form of cost functions, they can run slower than the simpler algorithms only based on the gradient. It is shown in [2] that under reasonable conditions, global convergence can be obtained for the steepest descent gradient method, Newton's methods and trust-region method in the framework of retractions.

1.2.4 Other Approaches

Problem (1) can also be solved by infeasible approaches such as various penalty, augmented Lagrangian, and SDP relaxation methods, which typically relax the constraints and penalize their violations and thus generate infeasible intermediate points. As the focus of this paper is a feasible method, we do not discuss infeasible methods in this paper.

1.3 Broad Applications

Minimization with respect to an orthogonal matrix X arises in many interesting problems, which can be widely found in [18,2,31,37,6,12,20,47,57,61,32]. Examples include linear and nonlinear eigenvalue problems, subspace tracking, low-rank matrix optimization, polynomial optimization, combinatorial optimization, sparse principal component analysis, electronic structures computations, etc. Let us briefly describe a few applications below.

1-bit compressive sensing. Compressive sensing (CS) [8] acquires a sparse signal of interest, not by taking many uniform samples, but rather by measuring a few incoherent linear projections followed by an optimization-based reconstruction. Instead of measuring the linear projections, 1-bit CS records just the signs of the linear projections, which can be done at very high speed by a simple comparator. Since there is no magnitude information in signs, [33] recovers the angle of the unknown u by minimizing its ℓ_1-norm subject to the normalization constraint $\|u\|_2 = 1$ using a constraint-preserving algorithm. Likewise, normalization constraints are widely used on variables that are angles, chromaticity, surface normals, flow directions, etc.

Low-rank matrix optimization. Semidefinite programming (SDP) is a useful tool for modeling many applications arising in combinatorial optimization, nonconvex quadratic programming, systems and control theory, statistics, etc. One class of SDP, including examples such as the maxcut SDP relaxation [10] and correlation matrix estimation [22,49], is to find a low-rank positive semidefinite matrix Y with $Y_{ii} = 1$, $i = 1, \cdots, n$. Since

a rank–p matrix Y can be expressed as $V^\top V$ for $V = [V_1, \ldots, V_n] \in \mathbb{R}^{p\times n}$, the original problem over Y can be transformed into one over V subject to spherical constraints $\|V_i\|_2 = 1$, $i = 1, \ldots, n$. This transform is also useful for solving SDP relaxations for binary integer programs.

Eigenvalue problems and subspace tracking. To compute p largest eigenvalues of a symmetric n-by-n matrix A, one can maximize $\mathrm{tr}(X^\top AX)$ with respect to $X \in \mathbb{R}^{n\times p}$ satisfying $X^\top X = I$. The solution X spans the same subspace as does the p eigenvectors associated with the p largest eigenvalues. To solve the generalized eigenvalue problem $Ax = \lambda Bx$, one can maximize $\mathrm{tr}(X^\top AX)$ subject to $X^\top BX = I$. Computing the principle invariant subspace of a symmetric or Hermitian matrix arises in adaptive filtering, direction finding, and other signal processing problems [46, 59]. For example, in a time-varying interference scenario (e.g., for airborne surveillance radar [56]), one computes the principal invariant subspace of $X_k = X_{k-1} + x_k x_k^\top$ at discrete times k. Since the new solution is close to the previous one, an optimization algorithm starting from the previous solution can be more efficient than the traditional eigenvalue and singular value decompositions.

Sparse principal component analysis (PCA). PCA methods such as [15, 58] are widely used is data analysis and dimension reduction. The sparse PCA problem is to find principal components consisting of only a few of the original variables while maintaining the good features of PCA such as uncorrelation of the principle components and orthogonality of the loading vectors. As an example, the recent work [34] introduces the model: $\min \rho|V| - \mathrm{Trace}(V^\top \Sigma V)$ subject to $V^\top V = I$ and $V^\top \Sigma V$ being diagonal.

Assignment problems. One of the fundamental combinatorial optimization problems is the assignment problem. In the QAP, there are a set of n facilities and a set of n locations, as well as a distance between each pair of facility and location. Given the flows between all the facility pairs, the problem is to assign the facilities to the locations to minimize the sum of the distance–flow products over all the pairs. These problems can be formulated as minimization over a permutation matrix $X \in \mathbb{R}^{n\times n}$, that is, $X^\top X = I$ and $X \geq 0$. It is well-known that the QAP is NP-hard in theory and difficult to solve even for moderately large n in practice.

The proposed algorithms were compared to the state-of-the-art solvers on some of the above problems and the numerical results are reported in Section 5. Note that some problems with additional constraints do not directly take the form of (3), yet they can be reduced to a sequence of subproblems in the form of (3).

1.4 Notation

A matrix $W \in \mathbb{R}^{n\times n}$ is called skew-symmetric (or anti-symmetric) if $W^\top = -W$. The trace of X, i.e., the sum of the diagonal elements of X, is denoted by $\mathrm{tr}(X)$. The Euclidean inner product between two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times n}$ is defined as $\langle A, B\rangle := \sum_{jk} A_{j,k}B_{j,k} = \mathrm{tr}(A^\top B)$. The Frobenius norm of $A \in \mathbb{R}^{m\times n}$ is defined as $\|A\|_F := \sqrt{\sum_{i,j} A_{i,j}^2}$. The set of $n \times n$ symmetric matrices is denoted by $\mathcal{S}^n$ and the set of $n \times n$ symmetric positive semidefinite (positive definite) matrices is denoted by $\mathcal{S}^n{}_+$ ($\mathcal{S}^n{}_{++}$). The notion $X \succeq 0$ ($X \succ 0$) means that the matrix $X \in S^n$ is positive semidefinite (positive definite). The notion $X \geq 0$ ($X > 0$) means that X is component-wise nonnegative (strictly positive). Given a differentiable function $\mathcal{F}(X) : \mathbb{R}^{n\times p} \to \mathbb{R}$, the gradient

of $\mathcal{F}$ with respect to X is denoted by $G := \mathcal{D}\mathcal{F}(X) := \left(\frac{\partial \mathcal{F}(X)}{\partial X_{i,j}}\right)$. The derivative of $\mathcal{F}$ at X in a direction Z is

$$\mathcal{D}\mathcal{F}(X)[Z] := \lim_{t\to 0} \frac{\mathcal{F}(X+tZ) - \mathcal{F}(X)}{t} = \langle \mathcal{D}\mathcal{F}(X), Z\rangle\,. \tag{9}$$

We reserve $\nabla\mathcal{F}$ for gradients in tangent planes.

1.5 Organization

The rest of this paper is organized as follows. Subsection 2.1 gives the optimality conditions of problem (1), and subsection 2.2 describes the proposed constraint-preserving scheme, as well as fast ways to compute it. A globally convergent monotone curvilinear search algorithm, whose step size is chosen according to the Armijo-Wolfe condition, is presented in subsection 3.1. A faster nonmonotone curvilinear search method with the Barzilai-Borwein step size is given in subsection 3.2. The proposed basic scheme is interpreted using the manifold concepts in subsection 4.1. Extensions to problems (2) and (3) are discussed in subsection 4.2. Global optimality for some special cases is discussed in subsection 4.3. Finally, numerical results on a variety of test problems are presented in section 5 to demonstrate the efficiency and robustness of the proposed algorithms.

2 Constraint-Preserving Update Schemes

2.1 Optimality Conditions

In this subsection, we state the first–order and second–order optimality conditions of problem (1). Since the matrix $X^\top X$ is symmetric, the Lagrangian multiplier Λ corresponding to $X^\top X = I$ is a symmetric matrix. The Lagrangian function of problem (1) is

$$\mathcal{L}(X,\Lambda) = \mathcal{F}(X) - \frac{1}{2}\mathrm{tr}(\Lambda(X^\top X - I)). \tag{10}$$

Lemma 1 *Suppose that X is a local minimizer of problem* (1)*. Then X satisfies the first-order optimality conditions $\mathcal{D}_X\mathcal{L}(X,\Lambda) = G - XG^\top X = 0$ and $X^\top X = I$ with the associated Lagrangian multiplier $\Lambda = G^\top X$. Define*

$$\nabla\mathcal{F}(X)^1 := G - XG^\top X \text{ and } A := GX^\top - XG^\top.$$

Then $\nabla\mathcal{F} = AX$. Moreover, $\nabla\mathcal{F} = 0$ if and only if $A = 0$.

Proof It follows from $X^\top X = I$ that the linear independence constraint qualification is satisfied. Hence, there exists a Lagrange multiplier Λ such that

$$\mathcal{D}_X\mathcal{L}(X,\Lambda) = G - X\Lambda = 0. \tag{11}$$

Multiplying both sides of (11) by $X^\top$ and using $X^\top X = I$, we have $\Lambda = X^\top G$. Since Λ must be a symmetric matrix, we obtain $\Lambda = G^\top X$ and $\mathcal{D}_X\mathcal{L}(X,\Lambda) = G - XG^\top X = 0$. Finally, it is easy to verify the last two statements.

[1] do not confuse $\nabla\mathcal{F}(X)$ with $G = \mathcal{D}\mathcal{F}(X)$

By differentiating both sides of $X^\top X = I$, we obtain the tangent vector set of the constraints, which is also the tangent space of $\mathcal{M}_n^p$ at X:

$$\mathcal{T}_X\mathcal{M}_n^p := \{Z \in \mathbb{R}^{n\times p} : X^\top Z + Z^\top X = 0\}. \tag{12}$$

Besides (12), another expression is $\mathcal{T}_X\mathcal{M}_n^p = \{X\Omega + X_\perp K : \Omega^\top = -\Omega, K \in \mathbb{R}^{(n-p)\times p}\}$, where $X_\perp \in \mathbb{R}^{n\times(n-p)}$ such that $XX^\top + X_\perp X_\perp^\top = I$. We can state the second-order optimality conditions as follows.

Lemma 2 *1) (Second-order necessary conditions, Theorem 12.5 in [40]) Suppose that $X \in \mathcal{M}_n^p$ is a local minimizer of problem* (1)*. Then X satisfies*

$$\mathrm{tr}\left(Z^\top \mathcal{D}(\mathcal{D}\mathcal{F}(X))[Z]\right) - \mathrm{tr}(\Lambda Z^\top Z) \geq 0, \quad \forall Z \in \mathcal{T}_X\mathcal{M}_n^p, \quad \text{where } \Lambda = G^\top X. \tag{13}$$

2) (Second-order sufficient conditions, Theorem 12.6 in [40]) Suppose that for $X \in \mathcal{M}_n^p$, there exists a Lagrange multiplier Λ such that the first–order conditions are satisfied. Suppose also that

$$\mathrm{tr}\left(Z^\top \mathcal{D}(\mathcal{D}\mathcal{F}(X))[Z]\right) - \mathrm{tr}(\Lambda Z^\top Z) > 0, \tag{14}$$

for any matrix $Z \in \mathcal{T}_X\mathcal{M}_n^p$. Then X is strict local minimizer for (1)*.*

2.2 The Update Scheme

The proposed update scheme is an adaptation of the classical steepest descent step to the orthogonality constraints, which are preserved at a reasonable computational cost. Since $\nabla\mathcal{F} = AX$ is the gradient of the Lagrangian function (also the steepest descent direction in the tangent plane at X; see section 4.1), a natural idea is to compute the next iterates as $Y = X - \tau AX$, where τ is a step size. The obstacle is that the new point Y may not satisfy $Y \in \mathcal{M}_n^p$. Using the similar technique as in [23,54], we modify the term $\nabla\mathcal{F}$ and compute the new iteration $Y_A(\tau)$ from the equations $Y_A(\tau) = X - \tau A\left(\frac{X+Y_A(\tau)}{2}\right)$. Replacing A by W just for the notation reason, the curve $Y_W(\tau)$ is given by

$$Y_W(\tau) = X - \tau W\left(\frac{X + Y_W(\tau)}{2}\right) \tag{15}$$

which satisfies $Y_W(\tau)^\top Y_W(\tau) = X^\top X$ for any skew-symmetric W and $\tau \in \mathbb{R}$. When there is no confusion, we omit the subscripts A and W and write both $Y_A(\tau)$ and $Y_W(\tau)$ as $Y(\tau)$. Lemma 3 below shows that the orthogonality constraints are preserved and $Y(\tau)$ defines a descent direction at $\tau = 0$.

Lemma 3 *1) Given any skew-symmetric matrix $W \in \mathbb{R}^{n\times n}$, then the matrix $Q := (I + W)^{-1}(I - W)$ is well-defined and orthogonal, i.e., $Q^\top Q = I$.*

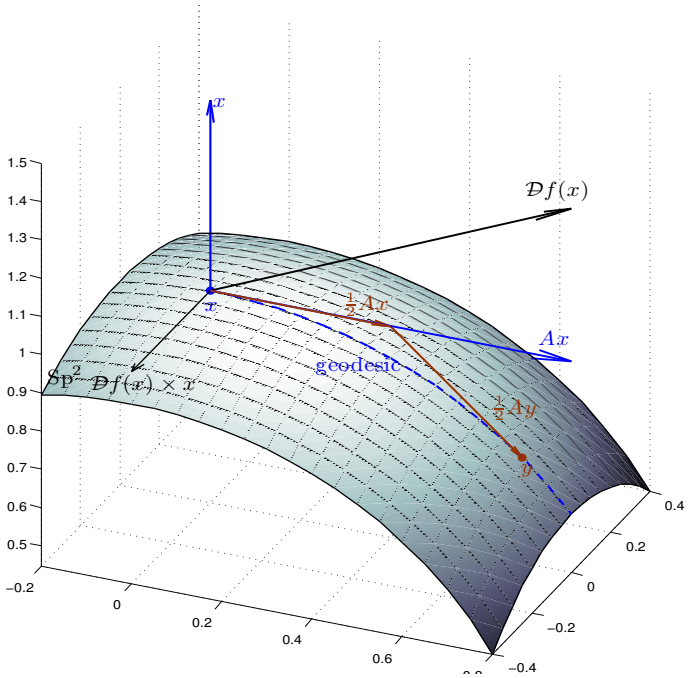
2) Given any skew-symmetric matrix $W \in \mathbb{R}^{n\times n}$, the matrix $Y(\tau)$ defined by (15) *satisfies $Y(\tau)^\top Y(\tau) = X^\top X$. $Y(\tau)$ can be expressed as*

$$Y(\tau) = \left(I + \frac{\tau}{2}W\right)^{-1}\left(I - \frac{\tau}{2}W\right)X, \tag{16}$$

and its derivative with respect to τ is

$$Y'(\tau) = -\left(I + \frac{\tau}{2}W\right)^{-1} W\left(\frac{X + Y(\tau)}{2}\right), \tag{17}$$

Fig. 1 An illustration using $\mathrm{Sp}^2 := \{x \in \mathbb{R}^3 : \|x\|_2 = 1\}$. The point $x + \tau Ax$ is not feasible. The curve $y(\tau)$ satisfying $y = x + \frac{\tau}{2}A(x + y)$ is feasible and a geodesic.



and, in particular, $Y'(0) = -WX$.

3) Set $W = A = GX^\top - XG^\top$. *Then* $Y(\tau)$ *is a descent curve at* $\tau = 0$, *that is,*

$$\mathcal{F}'_\tau(Y(0)) := \frac{\partial \mathcal{F}(Y(\tau))}{\partial \tau}\Big|_{\tau=0} = -\frac{1}{2}\|A\|_F^2. \tag{18}$$

For simplicity, we let $\mathcal{F}'_\tau(Y(\tau))$ denote the derivative of $\mathcal{F}(Y(\tau))$ with respect to τ.

Proof Part 1): The proof of this part is well-known; see [24], for example. We give it for the sake of completeness. Since W is skew symmetric, $v^\top(I + W)v = \|v\|_2^2$ holds for any nonzero $v \in \mathbb{R}^n$; hence, $(I + W)$ is invertible and Q is well-defined. Using the skew-symmetry and $(I + B)(I - B) = (I - B)(I + B)$ for any matrix B, we obtain $Q^\top Q = (I + W)\,(I + W)^{-1}\,(I - W)^{-1}\,(I - W) = I$.

Part 2): From part 1) and (15), we obtain (16). Since $\frac{\tau}{2}W$ is skew-symmetric, part 1) gives $Y(\tau)^\top Y(\tau) = X^\top X$. Differentiating both sides of (15) with respect to τ, we obtain $Y'(\tau) = -W\left(\frac{X+Y(\tau)}{2}\right) - \frac{\tau}{2}WY'(\tau)$, which gives (17).

Part 3): Using the chain rule, we obtain $\mathcal{F}'_\tau(Y(\tau)) = \mathrm{tr}\left(\mathcal{D}\mathcal{F}(Y(\tau))^\top Y'(\tau)\right)$. At $\tau = 0$, $\mathcal{D}\mathcal{F}(Y(0)) = G$ and $Y'(0) = -(GX^\top - XG^\top)X$. Therefore, $\mathcal{F}'_\tau(Y(0)) = -\mathrm{tr}(G^\top(GX^\top - XG^\top)X) = -\frac{1}{2}\|A\|_F^2$.

The matrix inverse $\left(I + \frac{\tau}{2}W\right)^{-1}$ dominates the computation of $Y(\tau)$ in (16). Next, we study a few fast ways to compute it. In particular, this inversion becomes very cheap when W is formed as the outer product of two low–rank matrices.

Lemma 4 *1) Suppose $W = LR^\top - RL^\top$, where $L, R \in \mathbb{R}^{n\times p}$. Rewrite $W = UV^\top$ for $U = [L,\ R]$ and $V = [R,\ -L]$. If $I + \frac{\tau}{2}V^\top U$ is invertible, then (16) is equivalent to*

$$Y(\tau) = X - \tau U\left(I + \frac{\tau}{2}V^\top U\right)^{-1} V^\top X. \tag{19}$$

2) The vector case: suppose $W = ab^\top - ba^\top$, where $a, b \in \mathbb{R}^n$. Then (16) is given explicitly by

$$y(\tau) = x - \beta_1(\tau)a - \beta_2(\tau)b, \tag{20}$$

where $\beta_1(\tau) = \tau\frac{x^\top b - \frac{\tau}{2}\left((a^\top b)(x^\top b) - (x^\top a)(b^\top b)\right)}{1-\left(\frac{\tau}{2}\right)^2(a^\top b)^2+\left(\frac{\tau}{2}\right)^2\|a\|_2^2\|b\|_2^2}$ and $\beta_2(\tau) = -\tau\frac{x^\top a + \frac{\tau}{2}\left((a^\top b)(x^\top a) - (a^\top a)(x^\top b)\right)}{1-\left(\frac{\tau}{2}\right)^2(a^\top b)^2+\left(\frac{\tau}{2}\right)^2\|a\|_2^2\|b\|_2^2}$.

Proof To $I + \frac{\tau}{2}W = I + \frac{\tau}{2}UV^\top$, we apply the SMW formula:

$$(B + \alpha UV^\top)^{-1} = B^{-1} - \alpha B^{-1}U(I + \alpha V^\top B^{-1}U)^{-1}V^\top B^{-1} \tag{21}$$

with $B = I$ and obtain $\left(I + \frac{\tau}{2}W\right)^{-1} = I - \frac{\tau}{2}U(I + \frac{\tau}{2}V^\top U)^{-1}V^\top$. With $I - \frac{\tau}{2}W = I - \frac{\tau}{2}UV^\top$, we have

$$\begin{aligned} Y(\tau) &= X - \frac{\tau}{2}U\left(\left(I + \frac{\tau}{2}V^\top U\right)^{-1}\left(I - \frac{\tau}{2}V^\top U\right) + I\right)V^\top X \\ &= X - \tau U\left(I + \frac{\tau}{2}V^\top U\right)^{-1}V^\top X. \end{aligned}$$

In case of $p = 1$, (20) implies that computing $y(\tau)$ reduces to essentially five inner products or even two inner products when $b = x$. Since the SMW formula can be numerically unstable, the feasibility of $Y(\tau)^\top Y(\tau) = I$ may deteriorate after a certain number of iterations and we restore it using a modified Gram-Schmidt process.

If $p \ll n$, inverting $I + V^\top U \in \mathbb{R}^{2p\times 2p}$ is much easier than inverting $(I + \frac{\tau}{2}W) \in \mathbb{R}^{n\times n}$, so (19) should used to compute $Y(\tau)$. When $W = GX^\top - XG^\top$, forming $V^\top U$ needs $2np^2$ flops, and $V^\top X$ is a part of $V^\top U$ so $V^\top X$ comes for free. The inversion $\left(I + \frac{\tau}{2}V^\top U\right)^{-1}$ takes $O(p^3)$, and the final assembly of $Y(\tau)$ takes another $2np^2 + O(p^3)$. Hence, the computation complexity of $Y(\tau)$ in (19) is $4np^2 + O(p^3)$. The work of updating $Y(\tau)$ for a different τ (during backtracking line search) is the inversion and final assembly, so it has a lower cost at $2np^2 + O(p^3)$. In comparison, the QR-based retraction [2] has a complexity of $4np^2$ since computing the gradient $\nabla\mathcal{F}(X)$ and obtaining the QR-decomposition need $2np^2$ each. Moving along the geodesic on Stiefel manifold is more expensive [18] at a complexity of $8np^2 + O(p^3)$ since, referring to Corollary 2.2 of [18], computing each one of H, $Y(t)$, A and QR there takes $2np^2$.

If $p \geq n/2$, then (19) has no advantage over (16). Nevertheless, one can still apply the SMW formula recursively though the computational efficiency of this approach is yet to be verified in practice. Suppose that W in part 1) of Lemma 4 can be decomposed as

$$W = \sum_{i=1}^{k} W^{(i)} \quad \text{for} \quad W^{(i)} := L^{(i)}(R^{(i)})^\top - R^{(i)}(L^{(i)})^\top,$$

where $L^{(i)}, R^{(i)} \in \mathbb{R}^{n\times p_i}$ and $\sum_{i=1}^k p_i = p$. First, the inverse of $B^{(1)} := I + \frac{\tau}{2}W^{(1)}$ can be computed using the SMW formula. Then the resulting $(B^{(1)})^{-1}$ can plugged in (21) as B for computing the inverse of $B^{(2)} := B^{(1)} + \frac{\tau}{2}W^{(2)}$. Iterate this process and finally $I + \frac{\tau}{2}W = B^{(k-1)} + \frac{\tau}{2}W^{(k)}$ can be inverted.

Since $Y(\tau)$ is easier to compute when W has lower rank, we show how to construct a rank–1 matrix W from $A = GX^\top - XG^\top$ and still obtain a descent path $Y(\tau)$.

Lemma 5 *Let $A = GX^\top - XG^\top$ and $W = G_{(q)}X_{(q)}^\top - X_{(q)}G_{(q)}^\top$, where q is a column index chosen as*

$$q := \arg\max_{i=1,\cdots,p} \left\{ \mathrm{tr}\left(XG^\top (G_{(i)}X_{(i)}^\top - X_{(i)}G_{(i)}^\top) \right) \right\}. \tag{22}$$

Then $Y(\tau)$ given by (16) *is a descent path, which satisfies*

$$\mathcal{F}_\tau'(Y(0)) \le -\frac{1}{2p}\|A\|_F^2. \tag{23}$$

Proof It follows from Lemma 3 and the outer product form of the matrix multiplication that

$$-\frac{1}{2}\|A\|_F^2 = -\mathrm{tr}(XG^\top(GX^\top - XG^\top)) = -\sum_{i=1}^{p} \mathrm{tr}\left(XG^\top (G_{(i)}X_{(i)}^\top - X_{(i)}G_{(i)}^\top) \right) \tag{24}$$

Hence, it must hold that $\mathrm{tr}\left(XG^\top (G_{(q)}X_{(q)}^\top - X_{(q)}G_{(q)}^\top) \right) > 0$, where q is defined in (22). Therefore, we have

$$-\frac{1}{2}\|A\|_F^2 \ge -p\ \mathrm{tr}\left(XG^\top (G_{(q)}qX_{(q)}^\top - X_{(q)}G_{(q)}^\top) \right).$$

Hence, (23) holds for the curve $Y(\tau)$ defined by (15) with $W = G_{(q)}X_{(q)}^\top - X_{(q)}G_{(q)}^\top$.

The above result can be easily extended to forming a rank–$2r$ matrix W from certain r columns of G and X and obtaining a descent path $Y(\tau)$. There are generally many choices of (low–rank) matrices W that will lead good search paths; however, we leave this exposition to the future.

3 Curvilinear search approaches

3.1 Monotone curvilinear search algorithm

In this section, we assume that the curve $Y(\tau)$ is generated by a skew-symmetric matrix W satisfying the following condition:

Condition 1 *The matrix W in* (15) *is continuous in X and satisfies*

$$\mathcal{F}_\tau'(Y(0)) \le -\sigma\|A\|_F^2, \tag{25}$$

where $\sigma > 0$ is a constant.

It is well known that the steepest descent method with a fixed step size may not converge. However, by choosing the step size wisely, convergence can be guaranteed and its speed can be accelerated without significantly increasing the cost at each iteration. At iteration k, one can choose a step size by minimizing $\mathcal{F}(Y_k(\tau))$ along the curve $Y_k(\tau)$ with respect to τ. Since finding its global minimizer is computationally expensive, one is usually satisfied with an approximate minimizer such as a τ_k satisfying the Armijo-Wolfe conditions [40, 52, 19]:

$$\mathcal{F}(Y_k(\tau_k)) \le \mathcal{F}(Y_k(0)) + \rho_1\tau_k\mathcal{F}_\tau'(Y_k(0)) \tag{26a}$$

$$\mathcal{F}_\tau'(Y_k(\tau_k)) \ge \rho_2\mathcal{F}_\tau'(Y_k(0)), \tag{26b}$$

Algorithm 1: A gradient descent method with curvilinear search

Given an initial point $X_0 \in \mathcal{M}_n^p$.
Initialization: Set $k \leftarrow 0$, $\epsilon \geq 0$ and $0 < \rho_1 < \rho_2 < 1$.
while *true* **do**
  Prepare. Generate A according to (4).
  Compute the step size τ_k. Call line search along the path $Y_k(\tau)$ defined by (15) for $W = A$ to obtain a step size τ_k that satisfies the Armijo-Wolfe conditions (26a) and (26b).
  Update. Set $X_{k+1} \leftarrow Y(\tau_k)$.
  Stopping check. If $\|\nabla \mathcal{F}_{k+1}\| \leq \epsilon$, then **STOP**; Otherwise, $k \leftarrow k+1$ and continue.

where $0 < \rho_1 < \rho_2 < 1$ are two parameters. To find τ_k satisfying (26a) and (26b), we refer to algorithms 3.2 and 3.3 in [40], which are based on interpolation and bisection. For a more detailed description of such strategies, see [36]. Our curvilinear search approach is given in Algorithm 1.

Since $\mathcal{F}(Y(\tau))$ is continuously differentiable and bounded from below, it is not difficult to prove that there exists a τ_k satisfying the Armijo-Wolfe conditions (26a) and (26b). Therefore, every iteration of Algorithm 1 is well defined. Formally, we have

Lemma 6 (Lemma 3.1 of [40]) *If $0 < \rho_1 < \rho_2 < 1$ and $\mathcal{F}'_\tau(Y_k(0)) < 0$, there exist nonempty intervals of step lengths satisfying the Armijo-Wolfe conditions* (26a) *and* (26b).

We now study the convergence properties of the sequence $\{X_k\}$ generated by Algorithm 1. Starting from $X_0 \in \mathcal{M}_n^p$, the sequence $\{X_k\}$ generated by Algorithm 1 stays in a compact set $\mathcal{X} = \{X \,|\, \mathcal{F}(x) \leq \mathcal{F}(X_0),\, X \in \mathcal{M}_n^p\}$, since $\mathcal{F}(X_k)$ decreases monotonically and $X_k \in \mathcal{M}_n^p$ for all k. Hence, there exists at least one accumulation point. If $\lim_{k\in K} \tau_k = 0$, it can verified that the sequence $\{Y_k(\tau_k)\}_{k\in K}$ defined by formula (15) satisfies

$$\lim_{k\in K} \|Y_k(\tau_k) - Y_k(0)\|_F = 0 \text{ and } \lim_{k\in K} \|Y'_k(\tau_k) - Y'_k(0)\|_F = 0.$$

Using the same proof as in [23], the following result shows that $\{\nabla \mathcal{F}_k\}$ converges to zero.

Theorem 2 *Suppose Condition 1 is satisfied for the sequence $\{X_k\}$ generated by Algorithm 1. Then*

$$\lim_{k\to\infty} \|\nabla \mathcal{F}(X_k)\|_F = 0. \tag{27}$$

In fact, as we point out in the introduction, the Cayley transform is a retraction. Hence, the convergence results in [2] regarding retractions apply directly to Algorithm 1.

3.2 Nonmonotone Line Search with the BB step size

While our steepest descent Algorithm 1 is extremely simple, the well-known Barzilai-Borwein (BB) step size [5] is often able to accelerate the gradient method at nearly no extra cost. Our numerical experience in [23] of minimizing the p-harmonic flow energy subject spherical constraints is that the BB step size can significantly reduce the total number of steepest descent iterations. Hence, we apply the BB step size for solving (1), which

we shall now describe. In Algorithm 1, instead of choosing a step size τ_k to satisfy (26a) and (26b), we simply set τ_k to either

$$\tau_{k,1} = \frac{\mathrm{tr}\left((S_{k-1})^\top S_{k-1}\right)}{|\mathrm{tr}\left((S^{n-1})^\top Y_{k-1}\right)|} \quad \text{or} \quad \tau_{k,2} = \frac{|\mathrm{tr}\left((S_{k-1})^\top Y_{k-1}\right)|}{\mathrm{tr}\left((Y_{k-1})^\top Y_{k-1}\right)}, \tag{28}$$

where $S_{k-1} = X_k - X_{k-1}$ and $Y_{k-1} = \nabla\mathcal{F}(X_k) - \nabla\mathcal{F}(X_{k-1})$.

Since the BB step size does not necessarily decrease the objective value at every iteration, it may invalidate convergence, but this issue can be solved by introducing a globalization technique, which guarantees global convergence by regulating the step sizes in (28) only occasionally; see [14,44]. We adopt a non-monotone line search method based on a strategy in [62]. Specifically, the new points are generated iteratively in the form $X_{k+1} := Y_k(\tau_k)$, where $\tau_k = \tau_{k,1}\delta^h$ or $\tau_k = \tau_{k,2}\delta^h$ and h is the smallest integer satisfying

$$\mathcal{F}(Y_k(\tau_k)) \le C_k + \rho_1\tau_k\mathcal{F}'(Y_k(0)), \tag{29}$$

where each reference value C_{k+1} is taken to be the convex combination of C_k and $\mathcal{F}(X_{k+1})$ as $C_{k+1} = (\eta Q_k C_k + \mathcal{F}(X_{k+1}))/Q_{k+1}$, where $Q_{k+1} = \eta Q_k + 1$ and $Q_0 = 1$.

Algorithm 2: A Curvilinear Search method with BB steps

```
Given X_0, set τ > 0, ρ_1, δ, η, ϵ ∈ (0,1), k = 0.
while ‖∇F(X_k)‖ > ϵ do
    while F(Y_k(τ)) ≥ C_k + ρ_1 τ F'(Y_k(0)) do
        τ ← δτ
    X_{k+1} ← Y_k(τ), Q_{k+1} ← ηQ_k + 1 and C_{k+1} ← (ηQ_k C_k + F(X_{k+1}))/Q_{k+1}.
    Set τ ← max(min(τ_{k+1,1}, τ_M), τ_m), k ← k + 1.
```

4 Extensions

4.1 The Manifold Interpretation

In this subsection, we interpret the constraints-preserving scheme (15) in the setting of Stiefel manifold $\mathcal{M}_n^p$ and extend it to taking any tangent directions. The basic analysis of this manifold can be found, for example, in [18]. The tangent space $\mathcal{T}_X\mathcal{M}_n^p$ of $\mathcal{M}_n^p$ at X is particularly important here. In order to measure the length of the tangent vectors and define "steepest", an inner product $\langle\cdot,\cdot\rangle_X$ for $\mathcal{T}_X\mathcal{M}_n^p$ needs to be defined. There are two commonly known metrics for $\mathcal{T}_X\mathcal{M}_n^p$: the Euclidean metric $\langle Z_1, Z_2\rangle_e := \mathrm{tr}(Z_1^\top Z_2)$ and the canonical metric

$$\langle Z_1, Z_2\rangle_c := \mathrm{tr}(Z_1^\top P_X Z_2) = \mathrm{tr}(Z_1^\top(I - \frac{1}{2}XX^\top)Z_2),$$

where $Z_1, Z_2 \in \mathcal{T}_X\mathcal{M}_n^p$.

Given a differentiable scalar function $\mathcal{F}$ on $\mathcal{M}_n^p$, the gradient of $\mathcal{F}$ at X is defined as the unique element of $\mathcal{T}_X\mathcal{M}_n^p$ that satisfies

$$\langle\nabla_X\mathcal{F}, \xi\rangle_X = \mathcal{D}\mathcal{F}(X)[\xi], \quad \forall\xi \in \mathcal{T}_X\mathcal{M}_n^p, \tag{30}$$

where $\mathcal{D}\mathcal{F}(X)[\xi]$ is defined in (9). The gradient under the Euclidean metric is

$$\nabla_e \mathcal{F} := X\text{skew}(X^\top G) + (1 - XX^\top)G, \tag{31}$$

and the gradient under canonical metric is

$$\nabla_c \mathcal{F} := G - XG^\top X = \nabla \mathcal{F}, \tag{32}$$

which is used throughout Sections 2 and 3. The two gradients are equivalent in the sense that $\nabla_c \mathcal{F} = (I + XX^\top)\nabla_e \mathcal{F}$ and the eigenvalues of $(I + XX^\top)$ are bounded between 1 and 2.

Given an arbitrary direction $D \in \mathcal{T}_X \mathcal{M}_n^p$, the curve $Y(\tau)$ defined by (15) with $W := DX^\top - XD^\top$ satisfies $Y'(0) = -(D - XD^\top X) \neq -D$. However, we can obtain $Y'(0) = -D$ by letting

$$W := P_X DX^\top - XD^\top P_X \tag{33}$$

(see (5) for P_X), which is still skew symmetric and thus preserves orthogonality. From the chain rule, $Y'(0) = -D$ and the definition (30), we obtain

$$\mathcal{F}'_\tau(Y(0)) = \text{tr}(G^\top Y'(0)) = -\text{tr}(G^\top D) = -\left\langle \nabla_X \mathcal{F}, D \right\rangle_X.$$

Therefore, D defines a descent curve as long as $\langle \nabla_X \mathcal{F}, D\rangle_X > 0$. As examples, we set D as $\nabla_e \mathcal{F}, \nabla_c \mathcal{F} \in \mathcal{T}_X \mathcal{M}_n^p$ each and, from (33), obtain

$$W = \begin{cases} P_X GX^\top - XG^\top P_X, & \text{if } D = \nabla_e \mathcal{F}, \\ GX^\top - XG^\top, & \text{if } D = \nabla_c \mathcal{F}, \end{cases}$$

which gives derivatives of $\mathcal{F}$ along $Y(\tau)$, respectively, as

$$\mathcal{F}'_\tau(Y(0)) = \begin{cases} -\left\langle \nabla_e \mathcal{F}, \nabla_e \mathcal{F} \right\rangle_e, & \text{if } D = \nabla_e \mathcal{F}, \\ -\left\langle \nabla_c \mathcal{F}, \nabla_c \mathcal{F} \right\rangle_c = -\frac{1}{2}\|W\|_F^2, & \text{if } D = \nabla_c \mathcal{F}. \end{cases}$$

This clarifies the two different definitions (4) and (5) of A as follows: the former and latter correspond to the canonical and Euclidean metrics, respectively. With minor changes, the results in Sections 2 and 3 apply to A defined in (5).

The formula (33) lets one move along any tangent direction D by defining W and its associated curve $Y(\tau)$. It is possible to choose D corresponding to conjugate gradient, Newton, and quasi-Newton directions on $\mathcal{M}_n^p$, but these expositions are outside the scope of this paper.

4.2 Generalizations

A matrix $W \in \mathbb{C}^{n\times n}$ is called skew-Hermitian if $W^* = -W$. The inner product of two complex matrix $L, R \in \mathbb{C}^{n\times p}$ is defined as

$$\text{Tr}(L^* R) = \text{tr}(\text{real}(L)^\top \text{real}(R)) + \text{tr}(\text{imag}(L)^\top \text{imag}(R)),$$

where $\text{real}(L)$ and $\text{imag}(L)$ are the real and imaginary parts of L. The results in Lemmas 1 and 3 for problem (1) similarly apply to the complex problem (2) as follows.

Lemma 7 *1) Given any skew-Hermitian matrix $W \in \mathbb{C}^{n\times n}$ and symmetric positive definite matrix $M \in \mathbb{R}^{n\times n}$, the matrix $Q := (I + WM)^{-1}(I - WM)$ is well-defined and $Q^*MQ = M$. For any $\tau \in \mathbb{R}$ the solution $Y(\tau)$ of the equations*

$$Y(\tau) = X - \frac{\tau}{2}WM\,(X + Y(\tau)) \tag{34}$$

satisfies $Y(\tau)^*MY(\tau) = X^*MX$, and $Y(\tau)$ can be expressed as

$$Y(\tau) = \left(I + \frac{\tau}{2}WM\right)^{-1}\left(I - \frac{\tau}{2}WM\right)X = \left(M + \frac{\tau}{2}MWM\right)^{-1}\left(M - \frac{\tau}{2}MWM\right)X. \tag{35}$$

Furthermore, $Y'(0) = -WMX$.

2) Suppose X is a local minimizer of (2). Then, X satisfies the first-order necessary optimality conditions: $\nabla\mathcal{F} := G - MXG^*XK^{-1} = 0$, which are equivalent to $AX = 0$, where $A := GX^*M - MXG^*$. Hence, the scheme (34) with $W := A$ defines a descent direction at $\tau = 0$ and $\mathcal{F}'_\tau(Y(0)) = -\frac{1}{2}\|A\|_F^2$.

Proof Since M is positive definite, we have

$$Q := (I + WM)^{-1}(I - WM) = (M + MWM)^{-1}(M - MWM).$$

Since $x^*(M + MWM)x = x^*Mx > 0$ for any nonzero $x \in \mathbb{C}^n$, $(M + MWM)^{-1}$ and thus Q are well-defined. By following steps analogous to those in statement 1 of Lemma 3, we obtain

$$Q^*MQ = (M + MWM)(M - MWM)^{-1}M(M + MWM)^{-1}(M - MWM) = M.$$

All other statements can be proved in a similar fashion as in Lemmas 1 and 3.

Since the variables $X_1, \ldots, X_q$ are not coupled in the constraints of problem (3), the results of Lemmas 1, 3 and 7 naturally extend to problem (3).

4.3 Discussions on Global Optimality

Although spherical and orthogonality constraints are nonconvex, there are simple cases in which any local minimizer is provably a global one. Hence, in these cases our algorithms return global solutions as long as the saddle points, if any, are avoided (e.g., by starting from a random point or applying random perturbations). In this subsection, we present the global optimality results for the p largest eigenvalue problem and problems corresponding to SDPs with constraints on the diagonal entries only. The proof is based on checking the first and second-order necessary conditions for local minimizers.

Consider the SDP problem

$$\max_{X\succeq 0} \quad \mathrm{tr}(CX), \;\; \text{s.t. } X_{ii} = 1, \quad i = 1, \cdots, n, \tag{36}$$

where C is a given symmetric matrix. As is reviewed in subsection 1.3, if the solution $\bar{X}$ of (36) has rank p, then through the decomposition [10] $X = V^\top V$ with $V := [V_1, \ldots, V_n] \in \mathbb{R}^{p\times n}$, we obtain the equivalent problem

$$\max_{V=[V_1,\ldots,V_n]} \mathrm{tr}(CV^\top V), \;\; \text{s.t. } \|V_i\| = 1,\; i = 1, \ldots, n. \tag{37}$$

Compared to (36), (37) has fewer variables but is nonconvex. Largely based on results in [10], we have:

Theorem 3 *There exists $\bar{p} \le n$ such that, if $p \ge \bar{p}$, any local minimizer $\bar{V}$ of (37) is globally optimal and $\bar{V}^\top \bar{V}$ solves (36). In particular, $\bar{p}$ can be taken as $(n+1) - \inf\{\mathrm{rank}(C+D) : D \text{ is diagonal}\}$ or n, whichever is smaller.*

Proof Problem (37) satisfies the linear constraint qualification. Hence, $\bar{V}$ satisfies the first-order optimality condition: $(C-\Lambda)\bar{V}^\top = 0$ with the Lagrange multiplier $\Lambda := \mathrm{diag}(C\bar{V}^\top \bar{V})$, as well as the second-order necessary condition: $\mathrm{tr}((C-\Lambda)Z^\top Z) \le 0$ for all $Z = [Z_1, \ldots, Z_n] \in \mathbb{R}^{p\times n}$ satisfying $Z_i^\top \bar{V}_i = 0$, $i = 1, \ldots, n$, where $\bar{V}_i$ denotes the ith column of $\bar{V}$.

If $\mathrm{rank}(\bar{V}) < p$, there exists $z \in \mathbb{R}^p$ such that $z^\top \bar{V}_i = 0$, $i = 1, \ldots, n$. Take an arbitrary $\alpha \in \mathbb{R}^n$, and let $Z_i = \alpha_i z$, $i = 1, \ldots, n$. The second-order condition above reduces to $(z^\top z)\alpha^\top (C - \Lambda)\alpha \le 0$ and thus $(C - \Lambda) \preceq 0$. From the strong duality for SDP, this result and the first-order condition above give the following optimality results: $(\bar{V}^\top \bar{V}, \mathrm{diag}(\Lambda))$ is a pair of optimal primal–dual solutions for (36) and thus $\bar{V}$ is globally optimal for (36). If $p = n$ and $\mathrm{rank}(\bar{V}) = n$, the first order condition means that $(C - \Lambda) = 0$ and thus $(C - \Lambda) \preceq 0$, which again give the above optimality results. This means that, as long as $p = n$, the optimality results always hold. Finally, from the first-order condition, we have $\mathrm{rank}(\bar{V}) \le n - \mathrm{rank}(C - \Lambda)$. Hence, if $p \ge \bar{p} = (n+1) - \inf\{\mathrm{rank}(C+D) : D \text{ is diagonal}\}$, then $p > \mathrm{rank}(\bar{V})$ and the above optimality results hold.

The result of this theorem obviously holds for SDPs with constraints $X_{ii} = b_i,\ i \in \Omega \subseteq \{1, \ldots n\}$, where b_i's are any given positive scalars. Problem (36) appears as the maxcut SDP relaxation, where C equals the negative of the graph weight matrix. For certain cographs, $\mathrm{rank}(C + D)$ can be derived; see [13].

Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$, it is known from [30] that the sum of the p largest eigenvalues of A equals $\max_{X\in\mathbb{R}^{n\times p}}\{\mathrm{tr}(X^\top AX) : X^\top X = I\}$. The first-order and second-order necessary conditions for a local minimizer X are, respectively,

$$AX - X\Lambda = 0,$$
$$\mathrm{tr}\left(Z^\top AZ\right) - \mathrm{tr}\left(\Lambda Z^\top Z\right) \le 0,\ \forall Z \in \mathbb{R}^{n\times p} \text{ satisfying } X^\top Z + Z^\top X = 0,$$

where $\Lambda = X^\top AX$ is the Lagrangian multiplier matrix. It can be shown that any X satisfying these conditions must span the same subspace as that spanned by the p eigenvectors corresponding to the p largest eigenvalues (counting multiples).

5 Numerical Results

In this section, we demonstrate the effectiveness of our approaches on a wide variety of test problems. We implemented both Algorithms 1 and 2 in MATLAB (Release 7.9.0). They use the search curve $Y(\tau)$ generated by the skew-symmetric matrix $W := GX^\top - XG^\top$. The Armijo–Wolfe step size in Algorithm 1 is determined by the code "DCSRCH" [36] with an initial step size of 10^{-2} and parameters $\rho_1 = 10^{-4}$ and $\rho_2 = 0.9$. Since Algorithm 2, which uses the BB step size and nonmonotone curvilinear search, appears to be more efficient in most test sets, we compare both algorithms on the first test set in subsection 5.2 and compare only Algorithm 2 with one or two state-of-the-art algorithms on problems in the remaining test sets except the quadratic assignment problem. All experiments were performed on a Lenovo Workstation with an Intel Xeon E5506 Processor with access to 5GB of RAM.

5.1 Termination Rules and Detection of Stagnation

Since convergence of first-order methods can slow down as the iterates approach a stationary point, it is critical to detect this slowdown and stop properly. In addition, it is tricky to correctly predict whether an algorithm is temporarily or permanently trapped in a region when its convergence speed has reduced. Hence, it is usually beneficial to have flexible termination rules. In our implementation, in addition to checking the norm of the gradient $\|\nabla\mathcal{F}(X_k)\|$, we also compute the relative changes of the two consecutive iterates and their corresponding objective function values:

$$\mathrm{tol}_k^x = \frac{\|X_k - X_{k+1}\|_F}{\sqrt{n}} \text{ and } \mathrm{tol}_k^f = \frac{\mathcal{F}(X_k) - \mathcal{F}(X_{k+1})}{|\mathcal{F}(X_k)| + 1}. \tag{38}$$

We let our algorithms run up to K iterations and stop them at iteration $k < K$ if $\|\nabla\mathcal{F}(X_k)\| \le \epsilon$, or $\mathrm{tol}_k^x \le \epsilon_x$ and $\mathrm{tol}_k^f \le \epsilon_f$, or

$$\mathrm{mean}([\mathrm{tol}_{k-\min(k,T)+1}^x, \ldots, \mathrm{tol}_k^x]) \le 10\epsilon_x \text{ and } \mathrm{mean}([\mathrm{tol}_{k-\min(k,T)+1}^f, \ldots, \mathrm{tol}_k^f]) \le 10\epsilon_f.$$

The defaults values of $\epsilon, \epsilon_x, \epsilon_f, T$ and K are 10^{-5}, 10^{-5}, 10^{-8}, 5 and 1000, respectively.

5.2 Distribution of Electrons on a Sphere

The first set of test problems, originating from Thomson's plum pudding model of the atomic nucleus, was obtained from the benchmark set COPS 3.0 [17] for nonlinearly constrained optimization. These problems seek the lowest energy configuration of n_p-point charges on a conducting sphere. The potential energy of n_p points $(x_i; y_i; z_i)$ is given by

$$\mathcal{F}(x, y, z) := \sum_{i=1}^{n_p-1} \sum_{j=i+1}^{n_p} \left((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2\right)^{-\frac{1}{2}}$$

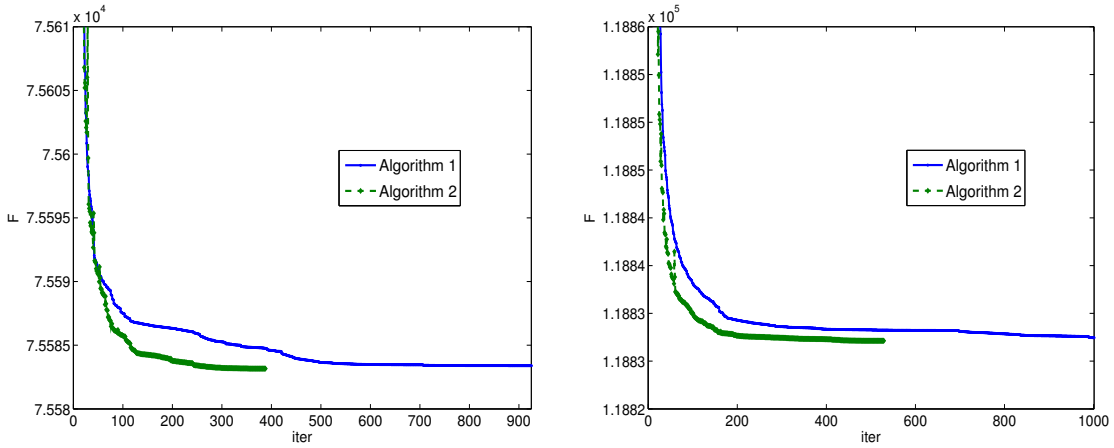
and the constraints over these n_p points are $x_i^2 + y_i^2 + z_i^2 = 1$, $i = 1, \ldots, n_p$.

We compared Algorithms 1 and 2 to the nonlinear programming software package "ipopt" [55] (version 3.8). The results are summarised in Table 1, where "$\mathcal{F}$", "feasi" and "nrmG" denote the final values of the objective function, constraint violation $\sqrt{\sum_i (\|(x_i; y_i; z_i)\| - 1)^2}$, and $\|\nabla\mathcal{F}(X)\|$, respectively, and "nfe" denotes the total number of function evaluations. The advantage of Algorithm 2 is large when $n_p \ge 100$. Note that the final objective values obtained by "ipopt" were slightly smaller than those by Algorithm 2 when $n = 400$ and 500, but the feasibility values of the "ipopt" solutions were much worse. Although "ipopt" took fewer function evaluations, as a second-order type method requiring expensive matrix factorizations, it spent more CPU time on the larger problems. Comparing Algorithm 1 with Algorithm 2, the latter took less CPU time and fewer function evaluations to achieve a similar solution accuracy. We depict their running objective values in Figures 2 (a) and (b), corresponding to $n = 400$ and $n = 500$, respectively. It is clear that Algorithm 2 converges faster than Algorithm 1 though the latter may sometimes achieve a slightly smaller objective function value.

Table 1 Computational summary for distribution of electrons on a sphere

n_p	50	100	200	300	400	500
IPOPT						
$\mathcal{F}$	1.055182e+03	4.448351e+03	1.843904e+04	4.213206e+04	7.558306e+04	1.188266e+05
feasi	1.968843e-07	2.517183e-07	4.160197e-07	5.003297e-07	5.923942e-07	5.999943e-07
nrmG	1.425497e-05	5.090285e-05	2.087299e-04	4.365843e-04	8.096863e-04	1.250355e-03
nfe	36	186	111	117	210	157
cpu	0.461	9.283	41.193	216.945	674.548	1244.886
Algorithm 1						
$\mathcal{F}$	1.055182e+03	4.448411e+03	1.843905e+04	4.213169e+04	7.558339e+04	1.188274e+05
feasi	5.087681e-16	6.181460e-16	9.614813e-16	9.992007e-16	1.280372e-15	1.447554e-15
nrmG	4.096349e-03	1.818747e-02	5.080282e-02	5.005146e-01	1.543954e-01	1.658408e+00
nfe	514	767	1070	1234	1338	1423
cpu	3.001	9.411	33.185	70.317	119.702	184.784
Algorithm 2						
$\mathcal{F}$	1.055182e+03	4.448351e+03	1.843906e+04	4.213177e+04	7.558316e+04	1.188271e+05
feasi	4.002966e-16	7.108896e-16	8.599751e-16	1.180183e-15	1.241267e-15	1.417438e-15
nrmG	1.213968e-03	5.441015e-03	2.660580e-02	2.950865e-02	8.700482e-02	9.229441e-02
nfe	151	242	331	229	418	558
cpu	0.779	2.906	10.140	12.915	37.016	71.741

Fig. 2 Objective function values versus iterations for examples in section 5.2



(a) n=400 (b) n=500

5.3 Application in Polynomial Optimization

In this subsection, we evaluate Algorithm 2 on polynomial optimization with spherical constraints, which arise from wide applications, see, [27,38]. Extensive research has been concentrated on relaxation techniques including the sum of squares (SOS) and Lasserres relaxations. Although the SOS relaxations and the associated SDPs have approximation bounds, their sizes are much larger than the original problems and thus become difficult to solve even when the original problems only have a few tens of variables. Here we apply Algorithm 2 to directly

minimizing homogeneous polynomials subject to spherical constraints. Note that unlike the SDP, Algorithm 2 has no approximation guarantees.

Four homogeneous polynomial problems listed in Table 2 were taken from the recent work [38], in which the resulted SDPs were solved by a specialized numerical method tailored from algorithms in [35, 63] and the corresponding objective function values f_{sos}^{hmg} are presented in Table 2. Since Algorithm 2 may reach different local minimizers, it was called multiple times from different random initial points, specifically, 10 and 1000 times for the first two and last two polynomials in Table 2, respectively. The minimal, mean and maximal final objective values, as well as the mean CPU seconds and feasibilities, are presented in the last three columns of Table 2. From the table, we can see that Algorithm 2 was able to find the same SDP solutions on three out of the four problems. While the SDP solver took more than a few hours on each of these problems in [38], the total time of Algorithm 2 counting the multiples runs is on the order of tens of seconds. We should point out that our approach and the SDP approach are not directly comparable due to different models.

Table 2 Polynomial optimization on a sphere

SDP		Algorithm 2		
$\mathcal{F}(x)$	f_{sos}^{hmg}	$\mathcal{F}(x)$ (min, mean, max)	mean CPU sec.	feasi
$\sum_{1\le i<j<k<l\le 50}(-i-j+k+l)x_ix_jx_kx_l$	-140.4051	(-140.4051,-140.4051,-140.4051)	2.45	8.0e-16
$\sum_{1\le i<j<k\le 49} x_ix_jx_k + x_i^2x_j - x_i^2x_k + x_jx_k^2$	-124.9645	(-124.9645,-124.9645,-124.9645)	0.10	3.8e-16
$\sum_{1\le i\le 20} x_i^6 + \sum_{1\le i\le 19} x_i^3x_{i+1}^3$	3.446e-5	(0.00006, 0.00143, 0.00391)	0.01	2.2e-15
$\sum_{1\le i<j<k\le 20} x_i^2x_j^2x_k^2 + x_i^3x_j^2x_k + x_i^2x_j^3x_k + x_ix_j^3x_k^2$	-0.3827	(-0.3827, -0.2725, -0.1059)	0.01	1.3e-15

The second set of problems compute stability numbers of graphs. Given a graph $G = (V, E)$, the stability number $\alpha(G)$ is the cardinality of the biggest stable subset of V (whose vertices are not connected to each other). It is shown in [38] that

$$\alpha(G)^{-1} = \min_{\|x\|_2=1} \sum_{i=1}^{n} x_i^4 + 2 \sum_{(i,j)\in E} x_i^2x_j^2.$$

We generated a few graphs G in the same way as [38] by first choosing a random subset $M \subset V$ with cardinality $n/2$ and then adding the edge $e_{i,j}$ with probability $\frac{1}{2}$ for each $\{i, j\} \not\subset M$. We ran Algorithm 2 from 10 different random points for each graph. The summary of performance is given in the last three columns of Table 3. For each graph, the best one of the 10 solutions of Algorithm 2 matches the SDP solution, yet the SDP algorithm in [38] took hours to run.

The third test set consists of homogeneous polynomial optimization problems with multiple spherical constraints from [27] in the form of

$$\max\ \mathcal{F}(x, y, z, w) := \sum_{1\le i\le n_1, 1\le j\le n_2, 1\le k\le n_3, 1\le l\le n_4} a_{ijkl}x_iy_jz_kw_l \ \text{ s.t. } \|x\|_2 = \|y\|_2 = \|z\|_2 = \|w\|_2 = 1,$$

where $A = (a_{ijkl})$ is a fourth-order tensor of size $n \times n \times n \times n$. In the test, A was generated randomly with i.i.d. standard Gaussian entries. We compared Algorithm 2 with the approximation algorithm "a1max4" proposed

Table 3 Stability number $\alpha(G)$

	Algorithm 2		
n	$\alpha(G)$	CPU (min, mean, max)	feasi
20	10	(0.004, 0.006, 0.007)	1.7e-15
30	15	(0.007, 0.009, 0.011)	7.8e-15
40	20	(0.005, 0.007, 0.010)	1.4e-15
50	25	(0.008, 0.010, 0.012)	5.5e-15
60	30	(0.009, 0.012, 0.015)	3.6e-15
80	40	(0.013, 0.015, 0.020)	2.0e-15
100	50	(0.018, 0.023, 0.036)	2.4e-15

in [27]. Since both algorithms are heuristics, they were called only once. Algorithm 2 was started from a random initial point. The main computation of "a1max4" is as cheap as calculating a few eigenvectors. From their performance reported in Table 4, we can see that Algorithm 2 was slower but returned higher (better) objective values than "a1max4".

Table 4 Numerical results on polynomial optimization with multiple spherical constraints

n	2	5	10	20	30	40	50	60	70
a1max4									
$\mathcal{F}$	3.253	5.668	7.930	10.662	11.995	13.498	16.029	17.305	18.161
feasi	8.429e-08	5.960e-08	1.788e-07	1.333e-07	2.920e-07	2.666e-07	1.686e-07	3.476e-07	2.065e-07
CPU	0.001	0.003	0.005	0.024	0.058	0.104	0.271	0.511	1.059
Algorithm 2									
$\mathcal{F}$	3.347	7.139	9.200	14.740	17.233	20.297	23.230	25.443	26.636
feasi	4.827e-15	1.064e-14	4.474e-15	1.898e-14	1.365e-14	2.150e-14	1.848e-14	2.443e-14	1.954e-14
nfe	55	112	105	116	125	133	141	144	153
CPU	0.185	0.381	0.900	3.595	10.843	25.723	53.291	97.453	172.634

5.4 Maxcut SDP Relaxation

Given a graph $G = (V, E)$ with $|V| = n$ and the weight matrix $W = (w_{ij})$, the maxcut problem partitions V into two nonempty sets $(S, V\backslash S)$ so that the total weights of the edges in the cut is maximized. This problem is NP-hard. For each node $i = 1, \ldots, n$, let $x_i = 1$ if $i \in S$ and, otherwise, $x_i = -1$. The maxcut problem can be formulated as

$$\max_x \ \frac{1}{2}\sum_{i<j} w_{ij}(1 - x_i x_j), \ \ \text{s.t. } x_i^2 = 1, \ i = 1, \ldots, n, \tag{39}$$

Relaxing the rank–1 matrix $xx^\top$ to a positive semidefinite matrix X and ignoring the rank–1 requirement, we obtain the SDP relaxation problem (36),where C is the Laplace matrix of the graph divided by 4, i.e.,

$C = -\frac{1}{4}(\mathrm{diag}(We) - W)$. Substituting the positive definite variable $X = V^\top V$ with $V := [V_1, \ldots, V_n] \in \mathbb{R}^{p\times n}$, we obtain problem (37) with multiple spherical constraints.

Theorem 3 means that with a mild p, the smaller problem (37) can yield the solution for (36), and a larger p is unnecessary. This low-rank structure has made the formulation (37) being popular and a few methods used it to construct cheap numerical methods for solving (36). The algorithm SDPLR [10] solves (37) using an augmented Lagrangian method, in which each subproblem is minimized by limited-memory BFGS iterations. Problem (37) can also be solved by EXPA [25], which forms an unconstrained differentiable exact penalty function and minimizes it using nonmontone Barzilai-Borwein gradient iterations. As opposed to our algorithms, these two algorithms do not preserve the spherical constraints during the iterations.

We compared Algorithm 2 to SDPLR's special maxcut version 0.130301 (written in the C-Language) on two test sets. EXPA was not tested since it is not publicly available. The first set includes four "torus" graphs in the DIMACS library. The second set includes all the "G"-set problems [7,28] with more than 2000 nodes, and these graphs were generated by "rudy", a machine independent graph generator by G.Rinaldi. The parameters of SDPLR were set according to the best parameter files "p.maxcut5" in its package. The dimension p of the vectors v_i in (37) was set to $\max(\min(\mathrm{round}(\sqrt{2n}/2), 20), 1)$ and the maximal number of iterations K was set to 600 for Algorithm 2. The comparison results are summarized in Table 5, where "obj" denotes the objective values returned by the solvers. From the table, we can see that Algorithm 2 solved most maxcut SDP relaxation problems more efficiently than SDPLR in terms of both CPU time and objective values. The CPU times of Algorithm 2 for the graphs from "G63" to "G81" did not increase too much may imply that these problem are relatively easy to solve.

5.5 Low-Rank Nearest Correlation Estimation

Let $C \in S^n$ be a given symmetric matrix and $H \in S^n$ a given nonnegative weight matrix. The rank constrained nearest correlation matrix problem estimates from C a matrix $X \in S^n_+$ with rank p or less as follows:

$$\min_{X\succeq 0} \ \frac{1}{2}\|H \odot (X - C)\|_F^2, \ X_{ii} = 1, \ i = 1, \ldots, n, \ \ \mathrm{rank}\ (X) \le p. \tag{40}$$

Many approaches have been proposed to solve (40); see the comprehensive literature reviews in [49,22].

Similar to the approach for the maxcut problem in section 5.4, we express the rank constraints $\mathrm{rank}\ (X) \le p$ explicitly as $X = V^\top V$ with $V = [V_1, \ldots, V_n] \in \mathbb{R}^{p\times n}$. Hence, (40) is reduced to the minimization of a quadratic polynomial over spheres as follows:

$$\min_{V\in\mathbb{R}^{p\times n}} \ \frac{1}{2}\|H \odot (V^\top V - C)\|_F^2, \ \ \text{s.t. } \|V_i\|_2 = 1, \ i = 1, \ldots, n. \tag{41}$$

The same decomposition $X = V^\top V$ are employed in the geometric optimization methods [26], majorization method [42] and trigonometric parametrization methods [45].

The test problems are Examples 5.1, 5.3 and 5.5 in [22] as follows:

Ex1: $n = 500$, the entries $G_{ij} = 0.5 + 0.5e^{-0.05|i-j|}$ for $i, j = 1, \ldots, n$. The weight matrix H is either the identity or a random matrix whose entries are uniformly distributed in [0.1,10] except for 2×100 entries in [0.01, 100].

Table 5 Computational results for the maxcut SDP relaxation on the torus and Gset problems. "T1" to "T4" denote the graphs 'torusg3-8', 'torusg3-15', 'toruspm3-8-50' and 'toruspm3-15-50', respectively.

SDPLR				Algorithm 2				
Name	n	obj	CPU	p	obj	CPU	nfe	feasi
T1	512	4.573576e+02	0.57	16	4.573580e+02	0.38	343	4.8e-15
T2	3375	3.134523e+03	20.32	20	3.134567e+03	8.46	625	1.2e-14
T3	512	5.278072e+02	0.45	16	5.278086e+02	0.24	236	4.7e-15
T4	3375	3.475072e+03	11.24	20	3.475131e+03	7.88	583	1.2e-14
G22	2000	1.413577e+04	9.31	20	1.413595e+04	2.10	300	1.0e-14
G23	2000	1.414519e+04	7.27	20	1.414551e+04	1.79	241	9.6e-15
G24	2000	1.414060e+04	9.23	20	1.414086e+04	1.78	240	9.3e-15
G25	2000	1.414406e+04	9.65	20	1.414425e+04	1.80	261	9.7e-15
G26	2000	1.413272e+04	6.34	20	1.413287e+04	1.47	220	9.3e-15
G27	2000	4.141531e+03	10.30	20	4.141659e+03	1.33	206	9.4e-15
G28	2000	4.100691e+03	11.46	20	4.100790e+03	2.23	332	9.5e-15
G29	2000	4.208853e+03	10.89	20	4.208889e+03	1.47	233	9.1e-15
G30	2000	4.215364e+03	17.34	20	4.215382e+03	1.86	249	9.6e-15
G31	2000	4.215444e+03	14.13	20	4.215462e+03	1.76	218	9.5e-15
G32	2000	1.567615e+03	3.69	20	1.567627e+03	3.55	635	9.6e-15
G33	2000	1.544291e+03	3.94	20	1.544296e+03	3.89	618	9.5e-15
G34	2000	1.546676e+03	3.72	20	1.546685e+03	3.90	623	8.8e-15
G35	2000	8.014551e+03	13.66	20	8.014737e+03	2.99	425	9.6e-15
G36	2000	8.005909e+03	18.64	20	8.005956e+03	4.60	617	9.6e-15
G37	2000	8.018327e+03	15.24	20	8.018621e+03	3.04	454	9.3e-15
G38	2000	8.014767e+03	13.44	20	8.014969e+03	2.97	445	9.4e-15
G39	2000	2.877588e+03	19.32	20	2.877644e+03	2.78	430	9.6e-15
G40	2000	2.864778e+03	12.94	20	2.864784e+03	2.49	369	9.7e-15
G41	2000	2.865195e+03	25.86	20	2.865215e+03	2.50	378	9.3e-15
G42	2000	2.946239e+03	21.21	20	2.946251e+03	2.70	433	9.3e-15
G48	3000	5.999956e+03	7.38	20	6.000000e+03	2.14	251	1.2e-14
G49	3000	5.999839e+03	5.26	20	6.000000e+03	2.07	240	1.1e-14
G50	3000	5.988114e+03	4.88	20	5.988172e+03	4.66	547	1.2e-14
G55	5000	1.103920e+04	14.72	20	1.103946e+04	7.75	407	1.5e-14
G57	5000	3.885369e+03	16.48	20	3.885403e+03	12.41	627	1.5e-14
G58	5000	2.013593e+04	90.80	20	2.013539e+04	13.63	620	1.5e-14
G60	7000	1.522191e+04	18.59	20	1.522224e+04	14.90	523	1.8e-14
G62	7000	5.430703e+03	23.19	20	5.430777e+03	16.96	623	1.7e-14
G63	7000	2.824334e+04	77.89	20	2.824284e+04	19.49	638	1.8e-14
G64	7000	1.046582e+04	128.52	20	1.046561e+04	20.19	644	1.7e-14
G65	8000	6.205298e+03	28.38	20	6.205384e+03	20.25	620	1.9e-14
G66	9000	7.076941e+03	33.58	20	7.077048e+03	22.02	624	2.0e-14
G67	10000	7.744093e+03	38.28	20	7.744265e+03	24.84	624	2.1e-14
G70	10000	9.861247e+03	52.12	20	9.861523e+03	24.78	626	2.1e-14
G72	10000	7.808215e+03	37.10	20	7.808381e+03	24.47	622	2.2e-14
G77	14000	1.104509e+04	51.25	20	1.104550e+04	33.10	636	2.5e-14
G81	20000	1.565514e+04	74.79	20	1.565574e+04	44.98	634	3.0e-14

Ex2: $n = 943$, G is based on 100,000 ratings for 1682 movies by 943 users from the Movielens data sets. The weight matrix H is either the identity or the one provided by T. Fushiki at Institute of Statistical Mathematics, Japan.

We compared Algorithm 2 with the majorization code Major [42] and the majorized penalty code PenCorr [22]. Their performance is presented in Table 6, where "resi" denotes the residual $\|H \odot (X - C)\|_F$ and "feasi" denotes the violation of $X_{ii} = 1$. The CPU time of PenCorr decreased as the rank p increased in most cases, but Algorithm 2 did the opposite. The advantage of latter algorithm becomes larger when the weight $H \neq I$. It is worth noting that the implementation of Major and thus its speed could be improved.

5.6 Linear Eigenvalue Problem

We next compute a few extreme eigenvalues and their corresponding eigenvectors. Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$ and an arbitrary unitary matrix $V \in \mathbb{R}^{n\times p}$, the trace of $V^\top AV$ is maximized when V is an orthogonal basis of the eigenspace associated with the p largest eigenvalues. Let $\lambda_1 \geq \ldots \geq \lambda_n$ be the eigenvalues of A. The p-largest eigenvalue problem can be formulated as

$$\sum_{i=1}^{p} \lambda_i := \max_{X\in\mathbb{R}^{n\times p}} \ \mathrm{tr}(X^\top AX) \ \text{ s.t. } X^\top X = I, \tag{42}$$

that is, $\mathcal{F}(X) = -\mathrm{tr}(X^\top AX)$.

We compared Algorithm 2 with the MATLAB function "eigs", which calls the Fortran library ARPACK, on two sets of positive definite matrices. It is worth noting that the performance of ARPACK through a direct call can be better than "eigs" due to Matlab's interface implementation. We, however, did not tune the interface since our goal is not to tell which is faster but to demonstrate the potential of Algorithm 2 as a general manifold algorithm. The first set includes a few randomly generated dense Wishart matrices assembled as $A = \bar{A}^\top \bar{A}$, where $\bar{A} \in \mathbb{R}^{n\times n}$ is a matrix whose elements are sampled from the standard Gaussian distribution. The results in Table 7 correspond to n varying from 500 through 5000 and a fixed $p = 6$ (hence, the objective value $\mathcal{F}$ equals the sum of 6 largest eigenvalues). In this table, "err" denotes the relative error between the objective values given by eigs and Algorithm 2, "nAx" denotes the total number of matrix-vector products in eigs, and "nfe" denotes the total number of function evaluations in Algorithm 2. We can see that our code works well when n is large. The results corresponding to varying p but fixed $n = 5000$ are presented in Tables 8, which show that Algorithm 2 performs well when p was relatively small. The second test set contains 39 large sparse matrices with $n \geq 4000$ from the UF Sparse Matrix Collection [16]. We tested computing two largest eigenvalues and the results are presented in Table 9, where, "resi" denotes the residual $\|AX - X\mathrm{diag}(\lambda_1, \ldots, \lambda_p)\|_F = \frac{1}{2}\|\nabla\mathcal{F}(X)\|_F$ corresponding to the computed eigenpairs. Since $\|\nabla\mathcal{F}(X)\|_F \leq \epsilon = 10^{-5}$ is one of our termination rules in section 5.1, only moderately accurate solutions were computed. We can see that Algorithm 2 was quite competitive on most problems in terms of $\mathcal{F}$, i.e., the summation of the first two eigenvalues, for achieving a residual on the order of 10^{-5}. The performance of both eigs and Algorithm 2 varied over different runs due to different initializations. It is worth noting that although eigs failed on problems "fv1" to "fv3" and "t2dal_e", it can become successful

Table 6 Computational results for the Low-Rank nearest correlation problem.

	Major			PenCorr			Algorithm 2			
p	resi	CPU	feasi	resi	CPU	feasi	resi	CPU	feasi	nfe
Ex1, $H = I$										
2	1.564201e+02	8.34	1.2e-13	1.564172e+02	53.20	4.3e-09	1.686832e+02	0.25	3.5e-15	42
5	7.883390e+01	4.54	1.4e-13	7.883423e+01	10.34	3.1e-08	7.882875e+01	1.12	3.7e-15	200
10	3.868525e+01	5.25	5.2e-15	3.868518e+01	6.30	2.9e-07	3.868258e+01	1.01	4.0e-15	182
20	1.570825e+01	12.64	5.6e-15	1.570796e+01	5.31	7.8e-08	1.570688e+01	1.26	5.1e-15	195
50	4.140789e+00	142.30	7.1e-15	4.139455e+00	2.29	5.2e-07	4.139235e+00	5.71	5.9e-15	533
100	1.471204e+00	928.58	9.5e-15	1.466498e+00	2.56	2.4e-07	1.467395e+00	19.46	8.0e-15	1076
125	1.055070e+00	1731.39	9.4e-15	1.048114e+00	2.85	3.0e-08	1.049154e+00	24.65	8.1e-15	1036
Ex1, random H										
2	9.106583e+02	14.64	1.4e-13	9.109902e+02	163.18	4.6e-07	9.778999e+02	0.76	3.6e-15	105
5	4.536099e+02	31.42	1.2e-13	4.537458e+02	91.15	7.7e-07	4.535966e+02	2.04	3.9e-15	281
10	2.204165e+02	67.21	5.1e-15	2.204421e+02	69.79	5.4e-07	2.204043e+02	2.08	4.2e-15	275
20	8.812054e+01	218.99	5.4e-15	8.812887e+01	69.65	3.6e-07	8.851307e+01	3.36	4.9e-15	381
50	2.203931e+01	2022.31	7.1e-15	2.191649e+01	94.46	8.6e-07	2.188864e+01	27.36	6.0e-15	2076
100	7.110542e+00	9649.30	9.2e-15	6.389784e+00	121.00	2.6e-07	6.457456e+00	39.97	8.0e-15	2074
125	5.030963e+00	13801.88	1.0e-14	4.179018e+00	151.09	9.6e-07	4.348972e+00	48.05	8.8e-15	2090
Ex2, $H = I$										
5	4.127677e+02	10.63	1.6e-13	4.128428e+02	172.70	6.5e-08	4.127542e+02	2.72	5.3e-15	162
10	3.265122e+02	46.72	6.2e-15	3.263352e+02	185.19	1.2e-07	3.262344e+02	2.73	5.6e-15	166
20	2.887315e+02	27.11	7.6e-15	2.887228e+02	88.19	3.2e-08	2.886782e+02	9.87	6.6e-15	488
50	2.763023e+02	78.99	1.0e-14	2.762742e+02	51.40	7.5e-08	2.762733e+02	11.74	8.8e-15	364
100	2.758067e+02	260.08	1.4e-14	2.757853e+02	9.44	1.7e-08	2.757854e+02	6.35	1.1e-14	121
150	2.758095e+02	925.63	1.7e-14	2.757853e+02	9.44	1.7e-08	2.757854e+02	7.40	1.3e-14	101
250	2.758087e+02	1676.32	2.1e-14	2.757853e+02	9.44	1.7e-08	2.757854e+02	11.11	1.6e-14	97
Ex2, H given by T. Fushiki										
5	1.141113e+04	407.47	2.6e-13	1.147849e+04	2480.31	5.7e-07	1.140483e+04	35.89	5.1e-15	1406
10	7.586522e+03	1299.88	6.1e-15	7.638036e+03	2305.20	2.8e-07	7.586921e+03	33.17	5.8e-15	1247
20	5.200671e+03	1733.57	7.3e-15	5.219117e+03	2062.14	2.7e-07	5.194958e+03	30.54	6.1e-15	1007
50	3.712916e+03	4154.29	9.0e-15	3.718507e+03	1356.16	2.0e-09	3.711896e+03	21.99	7.8e-15	512
100	3.503209e+03	6426.36	1.2e-14	3.507128e+03	906.02	3.3e-07	3.502541e+03	22.05	9.7e-15	346
150	3.501124e+03	8734.17	1.6e-14	3.505351e+03	824.69	3.1e-07	3.500676e+03	27.78	1.1e-14	324
250	3.501170e+03	15595.92	2.0e-14	3.505351e+03	864.67	3.1e-07	3.500654e+03	40.59	1.4e-14	312

after using a different set of carefully chosen parameters. Moreover, the matrices "crystm01" to "crystm03", "Muu" and "fv1" to "fv3" are perhaps not among those one would compute the leading eigenvalues since they have most of their eigenvalues clustered together, i.e., the leading eigenvalues do not clearly stand out. The advantage Algorithm 2 is more obvious if only the largest eigenvalue is computed while this advantage fades as p grows. Although "nfe" of Algorithm 2 was smaller than "nAx" of eigs on most of these problems in both sets, Algorithm 2 needs matrix multiplications like AX, $X \in \mathbb{R}^{n\times p}$, which is p-times more expensive than the

matrix-vector multiplications Ax, $x \in \mathbb{R}^n$, used by eigs. We expect to improve the performance of Algorithm 2 for eigenvalue problems by making use of advanced eigenvalue techniques in linear algebra.

Table 7 Eigenvalues on randomly generated dense matrices for fixed $p = 6$

n	500	1000	2000	3000	4000	5000
eigs						
$\mathcal{F}$	1.153e+04	2.316e+04	4.717e+04	7.075e+04	9.469e+04	1.187e+05
feasi	5.972e-15	8.545e-15	4.295e-15	9.169e-15	6.618e-15	7.112e-15
nAx	132	202	248	354	290	335
cpu	0.113	0.500	2.132	6.040	9.008	16.052
Algorithm 2						
$\mathcal{F}$	1.153e+04	2.316e+04	4.717e+04	7.075e+04	9.469e+04	1.187e+05
feasi	6.468e-16	9.314e-16	8.921e-16	1.073e-15	1.419e-15	1.852e-15
nfe	58	43	74	59	67	84
cpu	0.127	0.410	2.030	3.408	7.094	13.727
err	1.255e-06	9.882e-07	4.649e-06	5.341e-06	4.936e-06	9.378e-06

Table 8 Eigenvalues of a randomly generated 5000-dimensional dense matrix

n	1	3	5	7	9	11	13	15
eigs								
$\mathcal{F}$	1.986e+04	5.940e+04	9.873e+04	1.379e+05	1.769e+05	2.157e+05	2.545e+05	2.932e+05
nAx	200	350	350	372	439	515	468	384
feasi	1.776e-15	3.739e-15	6.531e-15	9.950e-15	8.200e-15	1.455e-14	1.337e-14	1.211e-14
cpu	9.281	16.405	16.133	17.274	20.355	23.849	22.183	17.768
Algorithm 2								
$\mathcal{F}$	1.986e+04	5.940e+04	9.873e+04	1.379e+05	1.769e+05	2.157e+05	2.545e+05	2.932e+05
feasi	4.441e-16	1.884e-15	1.587e-15	2.047e-15	1.740e-15	1.361e-15	2.384e-15	3.537e-15
nfe	91	104	94	98	102	92	101	131
cpu	4.148	13.550	14.867	16.016	17.843	16.714	19.711	27.867
err	1.325e-06	2.102e-07	1.431e-07	1.438e-06	5.221e-06	4.307e-06	2.292e-07	1.053e-08

5.7 Total Energy Minimization in Electronic Structure Calculation

We next test a class of nonlinear eigenvalue problem called Kohn-Sham equations. The continuous problem was first discretized by expressing a single electron wavefunction $\psi(r)$ as a linear combination of planewaves $\psi(r) = \sum_{j=1}^{n_g} c_j e^{ig_j^\top r}$, where $g_j \in \mathbb{R}^3$, $j = 1, \cdots, n_g$ are frequency vectors arranged in a lexicographical order. Then the uniformly sampled $\psi(r)$ can be denoted by a vector $x \in \mathbb{C}^n$ with $c = Fx$, where F is the discretized Fourier transformation matrix. Let $X \in \mathbb{C}^{n\times p}$ be a matrix that contains p discretized wavefunctions. The

Table 9 Computing the first two largest eigenvalues of 39 instances in the UF Sparse Matrix Collection

		eigs				Algorithm 2				
Name	n	$\mathcal{F}$	CPU	nAx	resi	$\mathcal{F}$	err	CPU	nfe	resi
mhd4800b	4800	4.392537e+00	0.041	54	1.3e-15	4.392537e+00	4.9e-09	0.021	15	6.9e-06
bcsstk36	23052	3.523673e+08	0.747	91	1.6e-15	3.523671e+08	4.5e-07	0.608	49	8.8e-05
bcsstk38	8032	8.042155e+11	0.055	20	7.4e-16	8.042155e+11	9.0e-14	0.045	12	3.5e-07
bcsstm39	46772	5.542200e+01	1.370	178	3.5e-15	5.542200e+01	1.6e-10	1.354	67	8.7e-06
crystm01	4875	1.061139e-11	0.143	126	1.7e-15	1.061139e-11	1.2e-19	0.064	35	2.8e-05
crystm02	13965	3.522817e-12	0.599	178	2.5e-15	3.522817e-12	3.5e-19	0.200	39	7.5e-05
crystm03	24696	1.957652e-12	1.242	195	2.7e-15	1.957651e-12	8.7e-20	0.359	37	5.7e-05
ct20stif	52329	1.773180e+12	0.714	38	4.3e-15	1.773180e+12	1.8e-10	1.514	44	1.4e-05
msc04515	4515	6.275788e+10	0.416	406	3.4e-15	6.275693e+10	1.5e-05	0.222	131	1.0e-04
msc10848	10848	1.185087e+12	0.264	38	2.5e-15	1.185087e+12	4.1e-09	0.279	36	2.0e-05
msc23052	23052	1.409469e+09	0.725	91	2.2e-15	1.409468e+09	1.1e-06	0.682	58	5.8e-05
pwtk	217918	2.016957e+08	181.390	1920	5.1e-15	2.016942e+08	7.2e-06	21.946	138	3.5e-04
sts4098	4098	5.099813e+08	0.021	20	1.1e-15	5.099813e+08	1.6e-10	0.062	43	5.9e-06
s1rmq4m1	5489	1.371517e+06	0.601	321	3.0e-15	1.371515e+06	2.0e-06	0.304	93	9.6e-05
s1rmt3m1	5489	1.930501e+06	0.592	341	4.6e-15	1.930499e+06	1.5e-06	0.229	94	2.1e-04
s2rmq4m1	5489	1.371527e+05	0.528	287	4.8e-15	1.371526e+05	3.6e-07	0.408	143	8.1e-05
s2rmt3m1	5489	1.930449e+05	0.579	341	3.4e-15	1.930449e+05	1.3e-08	0.242	102	1.7e-05
s3rmq4m1	5489	1.371619e+04	0.603	322	2.4e-15	1.371617e+04	1.4e-06	0.244	92	8.2e-05
s3rmt3m1	5489	1.930698e+04	0.581	341	5.0e-15	1.930698e+04	3.4e-07	0.240	101	4.1e-05
s3rmt3m3	5357	1.916587e+04	0.452	270	6.1e-15	1.916585e+04	1.4e-06	0.166	70	9.6e-05
ex15	6867	2.218375e+10	0.190	145	2.1e-15	2.218375e+10	1.9e-07	0.192	89	5.9e-05
bcsstk16	4884	9.028529e+09	0.146	73	2.5e-15	9.028529e+09	3.9e-09	0.070	27	4.6e-05
bcsstk17	10974	2.470064e+10	0.177	56	9.1e-16	2.470063e+10	1.4e-08	0.138	30	3.2e-05
bcsstk18	11948	7.723856e+10	0.131	55	1.9e-15	7.723856e+10	3.1e-09	0.105	28	1.9e-05
bcsstk25	15439	2.120041e+15	0.174	55	2.9e-15	2.120041e+15	1.5e-11	0.101	19	3.0e-06
bcsstk28	4410	1.244298e+09	0.036	20	7.1e-16	1.244298e+09	1.6e-10	0.049	22	5.5e-06
bcsstm25	15439	1.395419e+09	0.247	161	2.8e-15	1.395419e+09	3.7e-09	0.092	23	6.3e-05
Kuu	7102	1.080637e+02	0.885	394	1.9e-15	1.080637e+02	2.2e-07	0.276	86	3.0e-05
Muu	7102	1.679791e-03	1.560	969	3.6e-15	1.679771e-03	2.0e-08	0.500	194	8.8e-05
finan512	74752	5.611798e+01	2.184	127	2.2e-15	5.611798e+01	3.0e-09	1.658	43	6.6e-06
nd3k	9000	2.541715e+02	2.859	214	2.5e-15	2.541715e+02	2.2e-08	1.203	72	1.4e-05
nasa4704	4704	4.132959e+08	0.045	38	5.7e-16	4.132959e+08	2.6e-10	0.027	15	1.4e-05
nasasrb	54870	5.296036e+09	5.285	272	2.3e-15	5.296036e+09	9.0e-08	2.688	78	1.9e-05
fv1	9604	0.000000e+00	24.078	17318	9.8e-01	9.019638e+00	9.0e+00	0.466	173	5.6e-05
fv2	9801	0.000000e+00	26.560	17678	9.8e-01	9.019546e+00	9.0e+00	0.478	163	6.0e-05
fv3	9801	0.000000e+00	26.144	17678	9.4e-01	7.999393e+00	8.0e+00	0.455	165	5.6e-05
t2dal_e	4257	0.000000e+00	3.725	7688	1.4e+00	4.144770e-05	4.1e-05	0.047	41	1.9e-05
aft01	8205	1.000000e+15	0.149	88	4.0e-16	1.000000e+15	8.4e-13	0.034	11	1.5e-07
cfd1	70656	1.358235e+01	4.774	215	3.7e-15	1.358228e+01	5.3e-06	4.260	97	3.8e-05

finite-dimensional approximation to the continuous total energy function is defined as

$$E_{total}(X) := \mathrm{tr}\left(X^*(\frac{1}{2}L + V_{ion})X\right) + \frac{1}{2}\rho^\top L^\dagger \rho + \rho^\top \epsilon_{xc}(\rho) + E_{Ewald} + E_{rep}. \tag{43}$$

Here, L is a finite dimensional representation of the Laplacian operator in the planewave basis which can be decomposed as $L = F^* D_g F$, and D_g is a diagonal matrix with $\|g_j\|^2$ on the diagonal. V_{ion} denotes the ionic pseduopotentials sampled on the suitably chosen Cartesian grid. For simplicity, we let the function $\rho(X) := \mathrm{diag}(XX^*)$ be denoted by ρ. The pseudo-inverse $L^\dagger$ is defined as $L^\dagger = F^* D_g^\dagger F$, where $D_g^\dagger$ is a diagonal matrix whose diagonal entries $d_j = \|g_j\|^{-2}$ if $g_j \neq 0$, otherwise $d_j = 0$. The term $\epsilon_{xc}(\rho)$ represents the exchange-correlation energy per particle in a uniform electron gas of density ρ. The term E_{rep} measures the degree of repulsiveness of the local pseudo-potential with a term that corresponds to the non-singular part of ion-ion potential energy and the detail of E_{Ewald} can be found in [60]. Hence, the discretized minimization problem is

$$\min\ E_{total}(X)\ \text{ s.t. } X^*X = I, \tag{44}$$

whose first-order optimality conditions are

$$H(X)X - X\Lambda = 0, \quad X^*X = I,$$

where $H(X) := \frac{1}{2}L + V_{ion} + \mathrm{diag}(L^\dagger\rho) + \mathrm{diag}(\mu_{xc}(\rho))$, where $\mu_{xc}(\rho) = d\epsilon_{xc}(\rho)/d\rho$ and Λ is the Lagrangian multiplier that is a symmetric matrix.

The implementation of the objective function and its gradient used in Algorithm 2 was based on a MATLAB toolbox KSSOLV [60]. We compared Algorithm 2 with the three methods provided in KSSOLV: the self-consistent field (SCF) iteration, a direct constrained minimization (DCM) algorithm, and a trust-region enabled DCM algorithm (TRDCM), on seven examples. The test results are presented in Table 10. Algorithm 2 was able to achieve the same objective values on most problems. However, its advantage in terms of speed is not clear since the most expensive task, that is the computation of the objective function values and gradients, was not optimized.

5.8 Quadratic Assignment Problem

Given matrices $A, B \in \mathbb{R}^{n\times n}$, the QAP [11] minimizes a quadratic function over a permutation matrix X as

$$\min_{X\in\mathbb{R}^{n\times n}} \mathrm{tr}(A^\top XBX^\top),\ \text{s.t. } X^\top X = I,\ X \geq 0. \tag{45}$$

It is well-known that the QAP is NP-hard and numerically challenging even for moderately large n in practice. Our purpose here is to demonstrate that Algorithm 2 can return high-quality QAP solutions in very short times. Since the entries of any feasible X are binary, we replace X by the equivalent $X \odot X$, where $\odot$ is Hadamard product, and solve the equivalent model

$$\min_{X\in\mathbb{R}^{n\times n}} \psi(X) := \mathrm{tr}(A^\top (X\odot X)B(X\odot X)^\top),\ \text{s.t. } X^\top X = I,\ X \geq 0. \tag{46}$$

instead of (45). From our limited numerical experience, we found that Algorithm 2 can often return better (local) solutions on (46) than (45).

Table 10 Numerical results on total energy minimization

sih4									
solver	$\mathcal{F}$	nrmG	feasi	cpu	solver	$\mathcal{F}$	nrmG	feasi	cpu
scf	-6.176928e+00	1.35e-06	2.34e-15	11.25	dcm	-6.176928e+00	8.96e-06	7.18e-06	13.27
trdcm	-6.176928e+00	9.74e-06	3.40e-15	13.62	Alg 2	-6.176928e+00	2.08e-05	2.28e-15	12.28
qdot									
scf	2.771133e+01	1.96e-01	4.14e-15	13.34	dcm	2.780181e+01	4.07e-01	1.85e-02	12.06
trdcm	2.772539e+01	3.64e-01	2.40e-15	11.71	Alg 2	2.770286e+01	7.58e-03	3.03e-10	25.21
ptnio									
scf	-2.234837e+02	9.26e-02	1.84e-14	179.20	dcm	8.711967e+01	3.02e+01	4.43e+00	144.28
trdcm	-2.265951e+02	1.41e-01	1.94e-14	171.38	Alg 2	-2.267828e+02	5.69e-03	5.66e-14	304.35
co2									
scf	-3.512440e+01	1.84e-06	2.99e-15	16.22	dcm	-1.384362e+01	2.70e+00	1.13e+00	16.93
trdcm	-3.512440e+01	1.20e-04	3.28e-15	17.13	Alg 2	-3.512440e+01	7.05e-06	2.21e-14	18.66
h2o									
scf	-1.644051e+01	8.97e-07	1.29e-15	10.16	dcm	-1.710034e+01	9.85e-01	1.32e+00	12.67
trdcm	-1.644051e+01	2.30e-05	2.22e-15	11.96	Alg 2	-1.644051e+01	1.79e-06	1.70e-14	14.89
hnco									
scf	-2.863466e+01	2.79e-06	3.68e-15	23.51	dcm	-1.039781e+01	2.99e+00	1.76e+00	23.81
trdcm	-2.863466e+01	1.11e-04	4.64e-15	22.43	Alg 2	-2.863466e+01	5.36e-06	5.46e-14	33.54
c2h6									
scf	-1.442049e+01	1.71e-06	3.83e-15	16.60	dcm	-1.178454e+01	1.46e+00	9.46e-03	17.21
trdcm	-1.442049e+01	8.34e-05	3.13e-15	19.42	Alg 2	-1.442049e+01	2.39e-06	7.34e-15	20.32

Unlike $X^\top X = I$, the nonnegative constraints $X \geq 0$ were not preserved by the iterations of Algorithm 2. Instead, the standard augmented Lagrangian framework [40] was applied to deal with them. The augmented Lagrangian function is

$$L_\mu(X, \Lambda) := \psi(X) + \sum_{i,j} \rho(X_{ij}, \Lambda_{ij}, \mu),$$

where $\mu > 0$ is the penalty parameter and

$$\rho(t, \sigma, \mu) := \begin{cases} -\sigma t + \frac{1}{2}\mu t^2, & \text{if } t - \frac{\sigma}{\mu} \leq 0, \\ -\frac{1}{2\mu}\sigma^2, & \text{otherwise.} \end{cases}$$

Starting from $\mu_0 > 0$, $\Lambda_0 = 0$, and an initial orthogonal matrix X_0, the augmented Lagrangian method iterates between solving a subproblem and updating μ and Λ. In the experiment, the kth subproblem

$$\min_{X \in \mathbb{R}^{n\times n}} L_{\mu_k}(X, \Lambda_k), \text{ s.t. } X^\top X = I \tag{47}$$

was solved by Algorithm 2, and the solution X_{k+1} was used to update

$$\Lambda_{k+1} := \max\left(\Lambda_k - \mu_k X_{k+1}, 0\right) \text{ and } \mu_{k+1} = 1.2\mu_k.$$

The entries of the final solution X^* were rounded to integers.

We tested the above method on the 136 instances in QAPLIB, a quadratic assignment problem library. Since it often returned local solutions and their qualities varied with the penalty parameter μ, we ran the above method from 40 different random initial points for $\mu_0 = 0.1, 1, 10$ each and report the best one among the 120 solutions for each instance. This procedure was also stopped if the best known or a better upper bound is achieved. It is worth noting that no permutation matrices could be taken as initial points since all of them satisfy the first-order optimality conditions and thus are stationary points. Starting from them would immediately stall the algorithm. The quality of the solution is measured by its relative gap to the best feasible solution given in QAPLIB:

$$\text{gap}\% = \left(\frac{\text{best upper bound} - \psi(X)}{\text{best upper bound}} \times 100 \right) \%,$$

as well as the feasibility violation:

$$\text{feasi} = \|X^\top X - I\|_F + \|\min(X, 0)\|_1.$$

The smallest gap found by our algorithm was less than 5% on all problems except the "chr"-family problems and "esc32a". Hence, in order to save space, we only present the computational results on problems whose best upper bounds are achieved and whose size n is greater than 80 in Tables 11 and 12, respectively. In these tables, "obj" denotes the best upper bound, "min gap" denotes the smallest gap achieved by our greedy procedure, and "μ" and "feasi" denote the corresponding penalty parameter and feasibility at this point, respectively. The numbers "med gap", "max gap" and "CPU" denote the median and maximal of gap among all generated local solutions, and the averaged CPU time measured in seconds, respectively. As we can see from Table 11, the greedy method is able to find the best upper bound for 38 problems, including larges problems such as "esc128", "lipa80b" and "lipa90b". For all 21 large problems with $n \geq 80$ reported in Table 12, our approach was able to identify feasible solutions whose gap is less than 3%. In fact, 15 of the best gaps were less than 1% and 16 of the worst gaps were only less than 5%. In particular, the best and worst gaps for the largest problem "tai256c" were 0.842 and 2.401, respectively. Our numerical results are promising even comparing these of the well-developed LP and SDP relaxations techniques and their variants in terms of speed. Most of the average time spending on each random instance is no more than half a minute or even less. The most expensive problem "tai256c" only took around 4.5 minute for each instance. Hence, the total time for all of the 120 random instances are still not large. Since the worst gap is often reasonable, the computational time can be saved if a reasonably good approximation is acceptable.

6 Conclusions

Spherical and orthogonal constraints appear in many important classes of optimization problems. In this paper, we study a feasible approach for these problems. The main contribution is the development of a constraint-preserving update with curvilinear search and careful implementation into an efficient algorithm. The update formula is analyzed in the Stiefel manifold and generalized to one that can move along any given tangent directions.

Table 11 QAPLIB: exact recovery

			Algorithm 2					
Name	n	obj	μ	min gap %	med gap %	max gap %	CPU	feasi
chr12b †	12	9742	1.0e+01	0.000	58.222	134.223	0.59	0
esc128 †	128	64	1.0e+00	0.000	7.812	15.625	22.01	0
esc16a †	16	68	1.0e+00	0.000	5.882	8.824	0.35	0
esc16b †	16	292	1.0e+00	0.000	0.343	0.685	0.71	0
esc16c †	16	160	1.0e+00	0.000	0.000	0.000	0.61	0
esc16d †	16	16	1.0e+00	0.000	12.500	12.500	0.26	0
esc16e †	16	28	1.0e+00	0.000	7.143	7.143	0.28	0
esc16f †	16	0	1.0e+00	0.000	0.000	0.000	0.01	0
esc16g †	16	26	1.0e+00	0.000	7.692	7.692	0.17	0
esc16h †	16	996	1.0e+00	0.000	0.000	0.000	0.83	0
esc16i †	16	14	1.0e+00	0.000	0.000	0.000	0.23	0
esc16j †	16	8	1.0e+00	0.000	12.500	25.000	0.12	0
esc32b †	32	168	1.0e+00	0.000	20.238	35.714	0.90	0
esc32c †	32	642	1.0e+00	0.000	0.000	0.000	1.76	0
esc32d †	32	200	1.0e+00	0.000	6.000	11.000	1.18	0
esc32e †	32	2	1.0e+00	0.000	0.000	0.000	0.69	0
esc32g †	32	6	1.0e+00	0.000	33.333	33.333	0.78	0
esc64a †	64	116	1.0e+00	0.000	0.000	0.000	5.75	0
had12 †	12	1652	1.0e+00	0.000	0.726	2.179	0.70	0
had14 †	14	2724	1.0e+00	0.000	0.404	1.762	0.81	0
had16 †	16	3720	1.0e+00	0.000	0.215	0.591	0.93	0
had18 †	18	5358	1.0e+00	0.000	0.336	0.859	1.19	0
had20 †	20	6922	1.0e-01	0.000	0.520	2.427	1.22	0
lipa20a †	20	3683	1.0e+00	0.000	2.675	3.204	1.18	0
lipa20b †	20	27076	1.0e+00	0.000	14.614	15.951	0.79	0
lipa30a †	30	13178	1.0e+00	0.000	1.829	2.254	2.34	0
lipa30b †	30	151426	1.0e+00	0.000	16.295	16.640	1.38	0
lipa40b †	40	476581	1.0e+00	0.000	9.576	18.493	2.26	0
lipa50b †	50	1210244	1.0e+00	0.000	0.000	0.000	0.55	0
lipa60b †	60	2520135	1.0e+00	0.000	19.244	19.889	7.63	0
lipa70b †	70	4603200	1.0e+00	0.000	10.108	20.217	6.43	0
lipa80b †	80	7763962	1.0e+00	0.000	21.205	21.550	15.08	0
lipa90b †	90	12490441	1.0e+01	0.000	21.430	22.119	21.08	0
nug12 †	12	578	1.0e+00	0.000	5.536	13.495	0.46	0
nug14 †	14	1014	1.0e-01	0.000	5.128	10.454	0.53	0
nug16a †	16	1610	1.0e+00	0.000	4.099	8.571	0.67	0
nug16b †	16	1240	1.0e+00	0.000	1.129	2.258	0.60	0
tai12a †	12	224416	1.0e+00	0.000	8.753	14.529	0.49	0

Table 12 QAPLIB: $n \geq 80$

			Algorithm 2						
Name	n	obj	μ	obj	min gap %	med gap %	max gap %	CPU	feasi
esc128 †	128	64	1.0e+00	64	0.000	7.812	15.625	22.01	0
lipa80a	80	253195	1.0e+00	254922	0.682	0.813	1.048	17.69	0
lipa80b †	80	7763962	1.0e+00	7763962	0.000	21.205	21.550	15.08	0
lipa90a	90	360630	1.0e+00	362890	0.627	0.732	0.933	23.82	0
lipa90b †	90	12490441	1.0e+01	12490441	0.000	21.430	22.119	21.08	0
sko100a	100	152002	1.0e+01	153210	0.795	1.601	2.751	22.59	0
sko100b	100	153890	1.0e-01	155048	0.752	1.598	2.611	22.44	0
sko100c	100	147862	1.0e-01	148992	0.764	1.824	2.903	22.52	0
sko100d	100	149576	1.0e+01	150810	0.825	1.579	2.474	22.37	0
sko100e	100	149150	1.0e-01	150436	0.862	1.820	3.002	22.76	0
sko100f	100	149036	1.0e+00	150392	0.910	1.621	2.558	22.84	0
sko81	81	90998	1.0e+01	92020	1.123	1.875	2.750	14.47	0
sko90	90	115534	1.0e+01	116506	0.841	1.695	2.820	18.04	0
tai100a	100	21052466	1.0e+00	21636148	2.773	3.342	4.068	26.46	0
tai100b	100	1185996137	1.0e+00	1206641180	1.741	5.642	9.388	51.31	0
tai150b	150	498896643	1.0e+01	508524940	1.930	3.199	4.523	118.96	0
tai256c	256	44759294	1.0e-01	45136316	0.842	1.352	2.401	270.49	0
tai80a	80	13515450	1.0e+01	13916084	2.964	3.629	4.637	15.10	0
tai80b	80	818415043	1.0e-01	824640287	0.761	4.859	8.515	29.33	0
tho150	150	8133398	1.0e+00	8230504	1.194	2.040	2.795	84.77	0
wil100	100	273038	1.0e+01	274048	0.370	0.776	1.102	22.24	0

The proposed algorithms compare favorably with state-of-the-art algorithms on a variety of test problems. In particular, they can provide multi-fold accelerations over SDP algorithms on several problems arising from polynomial optimization, combinatorial optimization and the correlation matrix estimation, thus extending our ability in solving difficult and large-scale instances of these problems. However, further research is still needed to develop approximation guarantees for the proposed algorithms.

In nonlinear programming, infeasible methods are sometimes good choices since they can perhaps avoid certain local minimizers. Our numerical results, however, suggest that on many problems, preserving spherical or orthogonality constraints does not cause local minimization. When local minimizers are unavoidable, its speed advantage offsets its potential drawbacks. It remains theoretically interesting to identify suitable conditions under which the constraint–preserving algorithms can indeed find the global minimizes with either full or high probability.

Acknowledgements

We would like to thank Yin Zhang, Xin Liu, and Shiqian Ma for the discussions on optimization conditions, Jiawang Nie for the discussions on polynomial optimization, Chao Yang for the discussions on the Kohn-Sham

equation, as well as Franz Rendl and Etienne de Klerk for their comments on QAPs. We would also like to thank Defeng Sun and Yan Gao for sharing their code PenCorr and their improvement for Major, as well as sharing the test data for the nearest correlation matrix problem. The authors are grateful to Adrian Lewis, the Associate Editor and three anonymous referees for their detailed and valuable comments and suggestions.

References

1. P.-A. Absil, C. G. Baker, and K. A. Gallivan, *Trust-region methods on Riemannian manifolds*, Found. Comput. Math., 7 (2007), pp. 303–330.
2. P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, Princeton, NJ, 2008.
3. R. L. Adler, J.-P. Dedieu, J. Y. Margulies, M. Martens, and M. Shub, *Newton's method on Riemannian manifolds and a geometric model for the human spine*, IMA J. Numer. Anal., 22 (2002), pp. 359–390.
4. C. G. Baker, P.-A. Absil, and K. A. Gallivan, *An implicit trust-region method on Riemannian manifolds*, IMA J. Numer. Anal., 28 (2008), pp. 665–689.
5. J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.
6. G. Bencteux, E. Cancés, W. W. Hager, and C. Le Bris, *Analysis of a quadratic programming decomposition algorithm*, SIAM J. Numer. Anal., 47 (2010), pp. 4517–4539.
7. S. J. Benson, Y. Ye, and X. Zhang, *Solving large-scale sparse semidefinite programs for combinatorial optimization*, SIAM J. Optim., 10 (2000), pp. 443–461.
8. P. Boufounos and R. Baraniuk, *1-bit compressive sensing*, Conf. on. Info. Science and Systems (CISS), Princeton, New Jersey,, (2008).
9. I. Brace and J. H. Manton, *An improved bfgs-on-manifold algorithm for computing weighted low rank approximations*, in the 17th International Symposium on Mathematical Theory of Networks and Systems, 2006, pp. 1735–1738.
10. S. Burer and R. D. C. Monteiro, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Math. Program., 95 (2003), pp. 329–357.
11. R. E. Burkard, S. E. Karisch, and F. Rendl, *QAPLIB—a quadratic assignment problem library*, J. Global Optim., 10 (1997), pp. 391–403.
12. E. Cancès, C. Le Bris, and P.-L. Lions, *Molecular simulation and related topics: some open mathematical problems*, Nonlinearity, 21 (2008), pp. T165–T176.
13. G. J. Chang, L.-H. Huang, and H.-G. Yeh, *On the rank of a cograph*, Linear Algebra Appl., 429 (2008), pp. 601–605.
14. Y.-H. Dai and R. Fletcher, *Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming*, Numer. Math., 100 (2005), pp. 21–47.
15. A. d'Aspremont, L. El Ghaoui, M. Jordan, and G. R. Lanckriet, *A direct formulation for sparse pca using semidefinite programming*, SIAM REVIEW, 49 (2007), pp. 434–448.
16. T. A. Davis, *The university of florida sparse matrix collection*, tech. report, University of Florida, 2010.
17. E. D. Dolan, J. J. Moré, and T. S. Munson, *Benchmarking optimization software with cops 3.0*, tech. report, Mathematics and Computer Science Division, Argonne National Laboratory, February 2004.
18. A. Edelman, T. A. Arias, and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 303–353.
19. R. Fletcher, *Practical methods of optimization*, A Wiley-Interscience Publication, John Wiley & Sons Ltd., Chichester, second ed., 1987.
20. J. B. Francisco, J. M. Martínez, L. Martínez, and F. Pisnitchenko, *Inexact restoration method for minimization problems arising in electronic structure calculations*, Computational Optimization and Applications, (2010).
21. S. Friedland, J. Nocedal, and M. L. Overton, *The formulation and analysis of numerical methods for inverse eigenvalue problems*, SIAM J. Numer. Anal., 24 (1987), pp. 634–667.

22. Y. Gao and D. Sun, *A majorized penalty approach for calibrating rank constrained correlation matrix problems*, tech. report, National University of Singapore, 2010.
23. D. Goldfarb, Z. Wen, and W. Yin, *A curvilinear search method for the p-harmonic flow on spheres*, SIAM Journal on Imaging Sciences, 2, pp. 84–109.
24. G. H. Golub and C. F. Van Loan, *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, third ed., 1996.
25. L. Grippo, L. Palagi, and V. Piccialli, *An unconstrained minimization method for solving low-rank sdp relaxations of the maxcut problem*, Mathematical Programming, (2009), pp. 1–28. 10.1007/s10107-009-0275-8.
26. I. Grubišić and R. Pietersz, *Efficient rank reduction of correlation matrices*, Linear Algebra Appl., 422 (2007), pp. 629–653.
27. S. He, Z. Li, and S. Zhang, *Approximation algorithms for homogeneous polynomial optimization with quadratic constraints*, Mathematical Programming, 125 (2010), pp. 353–383.
28. C. Helmberg and F. Rendl, *A spectral bundle method for semidefinite programming*, SIAM J. Optim., 10 (2000), pp. 673–696.
29. U. Helmke and J. B. Moore, *Optimization and dynamical systems*, Communications and Control Engineering Series, Springer-Verlag London Ltd., London, 1994. With a foreword by R. Brockett.
30. R. A. Horn and C. R. Johnson, *Matrix analysis*, Cambridge University Press, Cambridge, 1985.
31. E. Kokiopoulou, J. Chen, and Y. Saad, *Trace optimization and eigenproblems in dimension reduction methods*, tech. report, University of Minnesota, 2010.
32. M. Kružík and A. Prohl, *Recent developments in the modeling, analysis, and numerics of ferromagnetism*, SIAM Rev., 48 (2006), pp. 439–483.
33. J. N. Laska, Z. Wen, W. Yin, and R. G. Baraniuk, *Trust, but verify: Fast and accurate signal recovering from 1-bit compressive measurements*, tech. report, Rice University, 2010.
34. Z. Lu and Y. Zhang, *An augmented lagrangian approach for sparse principal component analysis*, arXiv:0907.2079, (2009).
35. J. Malick, J. Povh, F. Rendl, and A. Wiegele, *Regularization methods for semidefinite programming*, SIAM Journal on Optimization, 20 (2009), pp. 336–356.
36. J. J. Moré and D. J. Thuente, *Line search algorithms with guaranteed sufficient decrease*, ACM Trans. Math. Software, 20 (1994), pp. 286–307.
37. A. Nemirovski, *Sums of random symmetric matrices and quadratic optimization under orthogonality constraints*, Math. Program., 109 (2007), pp. 283–317.
38. J. Nie, *Regularization methods for sum of squares relaxations in large scale polynomial optimization*, tech. report, Department of Mathematics, UCSD, 2009.
39. Y. Nishimori and S. Akaho, *Learning algorithms utilizing quasi-geodesic flows on the stiefel manifold*, Neurocomput., 67 (2005), pp. 106–135.
40. J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer, New York, second ed., 2006.
41. B. Owren and B. Welfert, *The Newton iteration on Lie groups*, BIT, 40 (2000), pp. 121–145.
42. R. Pietersz and P. J. F. Groenen, *Rank reduction of correlation matrices by majorization*, Quant. Finance, 4 (2004), pp. 649–662.
43. C. Qi, K. A. Gallivan, and P.-A. Absil, *Riemannian bfgs algorithm with applications*, in Recent Advances in Optimization and its Applications in Engineering, M. Diehl, F. Glineur, E. Jarlebring, and W. Michiels, eds., Springer Berlin Heidelberg, 2010, pp. 183–192.
44. M. Raydan, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM J. Optim., 7 (1997), pp. 26–33.
45. R. Rebonato and P. Jäckel, *The most general methodology to create a valid correlation matrix for risk management and option pricing purposes*, The Journal of Risk, 2 (1999), pp. 17–27.
46. R. Roy and T. Kailath, *Esprit — estimation of signal parameters via rotational invariance techniques*, IEEE Trans. Acoust., Speech, Signal Processing, (1989), pp. 984–995.

47. R. Schneider, T. Rohwedder, A. Neelov, and J. Blauert, *Direct minimization for calculating invariant subspaces in density functional computations of the electronic structure*, J. Comput. Math., 27 (2009), pp. 360–387.
48. M. Shub, *Some remarks on dynamical systems and numerical analysis*, in Dynamical systems and partial differential equations (Caracas, 1984), Univ. Simon Bolivar, Caracas, 1986, pp. 69–91.
49. D. Simon and J. Abell, *A majorization algorithm for constrained correlation matrix approximation*, Linear Algebra Appl., 432 (2010), pp. 1152–1164.
50. S. T. Smith, *Geometric optimization methods for adaptive filtering*, ProQuest LLC, Ann Arbor, MI, 1993. Thesis (Ph.D.)–Harvard University.
51. S. T. Smith, *Optimization techniques on Riemannian manifolds*, in Hamiltonian and gradient flows, algorithms and control, vol. 3 of Fields Inst. Commun., Amer. Math. Soc., Providence, RI, 1994, pp. 113–136.
52. W. Sun and Y.-X. Yuan, *Optimization Theory and Methods*, vol. 1 of Springer Optimization and Its Applications, Springer, New York, 2006. Nonlinear programming.
53. C. Udrişte, *Convex functions and optimization methods on Riemannian manifolds*, vol. 297 of Mathematics and its Applications, Kluwer Academic Publishers Group, Dordrecht, 1994.
54. L. A. Vese and S. J. Osher, *Numerical methods for p-harmonic flows and applications to image processing*, SIAM J. Numer. Anal., 40 (2002), pp. 2085–2104.
55. A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Math. Program., 106 (2006), pp. 25–57.
56. J. Ward, *Space-time adaptive processing for airborne radar*, Technical report of MIT, (1994).
57. V. Weber, J. VandeVondele, J. Hütter, and A. M. Niklasson, *Direct energy functional minimization under orthogonality constraints*, The Journal of Chemical Physics, 128 (2008).
58. D. M. Witten, R. Tibshirani, and T. Hastie, *A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis*, Biostatistics, 10 (2009), pp. 515–534.
59. B. Yang, *Projection approximation subspace tracking*, IEEE TRANSACTIONS ON SIGNAL PROCESSING, 43 (1995), pp. 95–.
60. C. Yang, J. C. Meza, B. Lee, and L.-W. Wang, *Kssolv—a matlab toolbox for solving the kohn-sham equations*, ACM Trans. Math. Softw., 36 (2009), pp. 1–35.
61. C. Yang, J. C. Meza, and L.-W. Wang, *A constrained optimization algorithm for total energy minimization in electronic structure calculations*, J. Comput. Phys., 217 (2006), pp. 709–721.
62. H. Zhang and W. W. Hager, *A nonmonotone line search technique and its application to unconstrained optimization*, SIAM J. Optim., 14 (2004), pp. 1043–1056.
63. X. Zhao, D. Sun, and K. Toh, *A newton-cg augmented lagrangian method for semidefinite programming*, SIAM Journal on Optimization, 20 (2010), pp. 1737–1765.