Solving log-determinant optimization problems by a Newton-CG primal proximal point algorithm

Chengjing Wang [*], Defeng Sun [†] and Kim-Chuan Toh [‡]

September 30, 2009; Revised, March 12, 2010

Abstract

We propose a Newton-CG primal proximal point algorithm for solving large scale log-determinant optimization problems. Our algorithm employs the essential ideas of the proximal point algorithm, the Newton method and the preconditioned conjugate gradient solver. When applying the Newton method to solve the inner sub-problem, we find that the log-determinant term plays the role of a smoothing term as in the traditional smoothing Newton technique. Focusing on the problem of maximum likelihood sparse estimation of a Gaussian graphical model, we demonstrate that our algorithm performs favorably comparing to the existing state-of-the-art algorithms and is much more preferred when a high quality solution is required for problems with many equality constraints.

Keywords: Log-determinant optimization problem, Sparse inverse covariance selection, Proximal point algorithm, Newton's method

[*]Department of Mathematics, National University of Singapore, 10 Lower Kent Ridge Road, Singapore 119076 (matwc@nus.edu.sg).

[†]Department of Mathematics and NUS Risk Management Institute, Department of Mathematics National University of Singapore, 10 Lower Kent Ridge Road, Singapore 119076 (matsundf@nus.edu.sg). The research of this author is partially supported by the Academic Research Fund Under Grant R-146-000-104-112.

[‡]Department of Mathematics National University of Singapore, 10 Lower Kent Ridge Road, Singapore 119076 (mattohkc@nus.edu.sg); and Singapore-MIT Alliance, 4 Engineering Drive 3, Singapore 117576.

1 Introduction

In this paper, by defining $\log 0 := -\infty$, we consider the following standard primal and dual log-determinant (log-det) problems:

$$(P) \qquad \min_X \{\langle C,\, X\rangle - \mu \log\det X \,:\, \mathcal{A}(X) = b,\;\; X \succeq 0\},$$

$$(D) \qquad \max_{y,Z} \{b^T y + \mu \log\det Z + n\mu(1 - \log\mu) \,:\, Z + \mathcal{A}^T y = C,\;\; Z \succeq 0\},$$

where $C \in \mathcal{S}^n$, $b \in \mathcal{R}^m$, $\mu \geq 0$ is a given parameter, $\mathcal{A} : \mathcal{S}^n \to \mathcal{R}^m$ is a linear map and $\mathcal{A}^T : \mathcal{R}^m \to \mathcal{S}^n$ is the adjoint of $\mathcal{A}$. We assume that $\mathcal{A}$ is surjective, and hence $\mathcal{A}\mathcal{A}^T$ is nonsingular. Note that the linear maps $\mathcal{A}$ and $\mathcal{A}^T$ can be expressed, respectively, as

$$\mathcal{A}(X) \;= \Big[\langle A_1,\, X\rangle, \ldots, \langle A_m,\, X\rangle\Big]^T, \quad \mathcal{A}^T(y) = \sum_{k=1}^m y_k A_k, \tag{1}$$

where $A_k,\; k = 1, \cdots, m$ are given matrices in $\mathcal{S}^n$. As for the explanation of all other main notations one may see Subsection 1.1.

It is clear that the log-det problem (P) is a convex optimization problem, i.e., the objective function $\langle C,\, X\rangle - \mu \log\det X$ is convex (on S^n_+), and the feasible region is convex. The log-det problems (P) and (D) can be considered as a generalization of linear semidefinite programming (SDP) problems. One can see that in the limiting case where $\mu = 0$, they reduce, respectively, to the standard primal and dual linear SDP problems. Log-det problems arise in many practical applications such as computational geometry, statistics, system identification, experiment design, and information and communication theory. Thus the algorithms we develop here can potentially find wide applications. One may refer to [5, 27, 24] for an extensive account of applications of the log-det problem.

For small and medium sized log-det problems, including linear SDP problems, it is widely accepted that interior-point methods (IPMs) with direct solvers are generally very efficient and robust; see for example [24, 25]. For log-det problems with m large and n moderate (say no more than $2,000$), the limitations faced by IPMs with direct solvers become very severe due to the need of computing, storing, and factorizing the $m \times m$ Schur matrices that are typically dense.

Recently, Zhao, Sun and Toh [32] proposed a Newton-CG augmented Lagrangian (NAL) method for solving linear SDP problems. This method can be very efficient when the problems are primal and dual nondegenerate. The NAL method is essentially a proximal point method applied to the primal problem where the inner sub-problems are solved by an inexact semi-smooth Newton method using a preconditioned conjugate gradient (PCG) solver. Recent studies conducted by Sun, Sun and Zhang [23] and Chan and Sun [6] revealed that under the constraint nondegenerate conditions for (P) and (D) (i.e., the primal and dual nondegeneracy conditions in the IPMs literature, e.g., [1]), the NAL method can locally be regarded as an approximate generalized Newton method applied

to a semismooth equation. The latter result may explain to a large extent why the NAL method can be very efficient.

As the log-det problem (P) is an extension of the primal linear SDP, it is natural for us to further use the NAL method developed for linear SDPs to solve log-det problems. Following what has been done in linear SDPs, our approach is to apply a Newton-CG primal proximal point algorithm (PPA) to (P), and then to use an inexact Newton method to solve the inner sub-problems by using a PCG solver to compute inexact Newton directions. We note that when solving the inner sub-problems in the NAL method for linear SDPs [32], a semi-smooth Newton method has to be used since the objective functions are differentiable but not twice continuously differentiable. But for log-det problems, the objective functions in the inner sub-problems are twice continuously differentiable (actually, analytic) due to the fact that the term $-\mu \log\det X$ acts as a smoothing term. This interesting phenomenon implies that the standard Newton method can be used to solve the inner sub-problem. It also reveals a close connection between adding the log-barrier term $-\mu\log\det X$ to a linear SDP and the technique of smoothing the KKT conditions [6].

In [19, 20], Rockafellar established a general theory on the global convergence and local linear rate of convergence of the sequence generated by the proximal point and augmented Lagrangian methods for solving convex optimization problems including (P) and (D). Borrowing Rockafellar's results, we can establish global convergence and local convergence rate for our Newton-CG PPA method for (P) and (D) without much difficulty.

In problem (P), we only deal with a matrix variable, but the PPA method we develop in this paper can easily be extended to more general log-det problems to include vector variables. Although this kind of extension seems trivial, however, the numerical experiments in Section 6 includes these problems, thus we still list them as follows:

$$\min\{\langle C,\, X\rangle - \mu \log\det X + c^T x - \nu \log x \,:\, \mathcal{A}(X) + Bx = b,\; X \succeq 0, x \geq 0\}, \quad (2)$$

$$\max\{b^T y + \mu \log\det Z + \nu \log z + \kappa \,:\, Z + \mathcal{A}^T y = C,\; z + B^T y = c,\; Z \succeq 0, z \geq 0\}, \quad (3)$$

where $\nu > 0$ is a given parameter, $c \in \mathcal{R}^l$ and $B \in \mathcal{R}^{m\times l}$ are given data, and $\kappa = n\mu(1-\log\mu) + l\nu(1-\log\nu)$.

In the implementation of our Newton-CG PPA method, we focus on the maximum likelihood sparse estimation of a Gaussian graphical model (GGM). This class of problems includes two subclasses. The first subclass is that the conditional independence of a model is *completely* known, and it can be formulated as follows:

$$\min_X \Big\{\langle S,\, X\rangle - \log\det X \,:\, X_{ij} = 0,\; \forall\,(i,j) \in \Omega,\; X \succeq 0\Big\}, \quad (4)$$

where Ω is the set of pairs of nodes (i,j) in a graph that are connected by an edge, and $S \in \mathcal{S}^n$ is a given sample covariance matrix. Problem (4) is also known as a sparse covariance selection problem. In [7], Dahl, Vandenberghe and Roychowdhury showed that when the underlying dependency graph is nearly-chordal, an inexact Newton method

combined with an appropriate PCG solver can be quite efficient in solving (4) with n up to $2,000$ but on very sparse data (for instance, when $n = 2,000$, the number of upper nonzeros is only about $4,000 \sim 6,000$). But for general large scale problems of the form (4), little research has been done in finding efficient algorithms to solve the problems. The second subclass of the GGM is that the conditional independence of the model is *partially* known, and it is formulated as follows:

$$\min_X \Big\{ \langle S,\, X\rangle - \log\det X + \sum_{(i,j)\notin\Omega} \rho_{ij}|X_{ij}| \,:\, X_{ij} = 0,\ \forall\,(i,j)\in\Omega,\, X \succeq 0 \Big\}. \tag{5}$$

In [8], d'Aspremont, Banerjee, and El Ghaoui, among the earliest, proposed to apply Nesterov's smooth approximation (NSA) scheme to solve (5) for the case where $\Omega = \emptyset$. Subsequently, Lu [11, 12] suggested an adaptive Nesterov's smooth (ANS) method to solve (5). The ANS method is currently one of the most effective methods for solving large scale problems (e.g., $n \geq 1,000$, $m \geq 500,000$) of the form (5). In the ANS method, the equality constraints in (5) are removed and included in the objective function via the penalty approach. The main idea in the ANS method is basically to apply a variant of Nesterov's smooth method [17] to solve the penalized problem subject to the single constraint $X \succeq 0$. In fact, both the ANS and NSA methods have the same principle idea, but the latter runs much slowly than the former. In contrast to IPMs, the greatest merit of the ANS method is that it needs much lower storage and computational cost per iteration. In [11], the ANS method has been demonstrated to be rather efficient in solving randomly generated problems of form (5), while obtaining solutions with low/moderate accuracy. However, as the ANS is a first-order method, it may require huge computing cost to obtain high accuracy solutions. In addition, as the penalty approach is used in the ANS method to solve (5), the number of iterations may increase drastically if the penalty parameter is updated frequently. Another limitation of the ANS method introduced in [11] is that it can only deal with the special equality constraints in (5). It appears to be difficult to extend the ANS method to deal with more general equality constraints of the form $\mathcal{A}(X) = b$.

After the first round of review, one referee brought [22] and [31] to our attention. These two papers also dealt with (5), but without equality constraints. According to our numerical experiments, for examples without equality constraints given in Section 6 (the real data), only the alternating direction method (ADM) proposed by Yuan [31] seems competitive with the PPA and the ANS [the ADM is slightly slower than the PPA, but faster than the ANS]. It is worth further investigating if the ADM approach of Yuan [31] can be used to solve the general model (5) efficiently.

Our numerical results show that for both problems (4) and (5), our Newton-CG PPA method can be very efficient and robust in solving large scale problems generated as in [8] and [11]. Indeed, we are able to solve sparse covariance selection problems with n up to $2,000$ and m up to 1.8×10^6 in about 26 minutes. For both problems, our method consistently outperforms the ANS method by a substantial margin, especially when the problems are large and the required accuracy tolerances are relatively high.

The remaining part of this paper is organized as follows. In Section 2, we give some preliminaries including a brief introduction on concepts related to the proximal point algorithm. In Section 3 and 4, we present the details of the PPA method and Newton-CG algorithm. In Section 5, we give the convergence analysis of our PPA method. The numerical performance of our algorithm is presented in Section 6. Finally, we give some concluding remarks in Section 7.

1.1 Notations

In this paper, all vectors are assumed to be finite dimensional. The symbols $\mathcal{R}^n$ denotes the n-dimensional Euclidean space. The set of all $m \times n$ matrices with real entries is denoted by $\mathcal{R}^{m\times n}$. The space of all symmetric $n \times n$ matrices is denoted by $\mathcal{S}^n$, $\langle \cdot, \cdot \rangle$ stands for the standard trace inner product in $\mathcal{S}^n$, and $\|\cdot\|$ denotes the Frobenius norm. Moreover, $\mathcal{S}^n_+$ (resp., $\mathcal{S}^n_{++}$) is the cone of $n \times n$ symmetric positive semidefinite (resp., definite) matrices. If $X \in \mathcal{S}^n_+$, we also write $X \succeq 0$. Given matrices X and Y in $\mathcal{R}^{m\times n}$, $X \circ Y$ denotes the Hadamard product of X and Y.

2 Preliminaries

For the sake of subsequent discussions, we first introduce some concepts related to the proximal point method based on the classic papers by Rockafellar [19, 20].

Let H be a real Hilbert space with an inner product $\langle \cdot, \cdot \rangle$. A multifunction $T : H \rightrightarrows H$ is said to be a *monotone operator* if

$$\langle z - z', w - w' \rangle \geq 0, \text{ whenever } w \in T(z),\ w' \in T(z'). \tag{6}$$

It is said to be *maximal monotone* if, in addition, the graph

$$G(T) = \{(z, w) \in H \times H |\ w \in T(z)\}$$

is not properly contained in the graph of any other monotone operator $T' : H \rightrightarrows H$. For example, if T is the subdifferential ∂f of a lower semicontinuous convex function $f : H \to (-\infty, +\infty], f \not\equiv +\infty$, then T is maximal monotone (see Minty [14] or Moreau [15]), and the relation $0 \in T(z)$ means that $f(z) = \min f$.

Rockafellar [19] studied a fundamental algorithm for solving

$$0 \in T(z), \tag{7}$$

in the case of an arbitrary maximal monotone operator T. The operator $P = (I + \lambda T)^{-1}$ is known to be single-valued from all of H into H, where $\lambda > 0$ is a given parameter. It is also *nonexpansive*:

$$\|P(z) - P(z')\| \leq \|z - z'\|,$$

and one has $P(z) = z$ if and only if $0 \in T(z)$. The operator P is called the *proximal mapping* associate with λT, following the terminology of Moreau [15] for the case of $T = \partial f$.

The PPA generates, for any starting point z^0, a sequence $\{z^k\}$ in H by the approximate rule:

$$z^{k+1} \approx (I + \lambda_k T)^{-1}(z^k).$$

Here $\{\lambda_k\}$ is a sequence of positive real numbers. In the case of $T = \partial f$, this procedure reduces to

$$z^{k+1} \approx \arg\min_z \Big\{ f(z) + \frac{1}{2\lambda_k}\|z - z_k\|^2 \Big\}. \tag{8}$$

Definition 2.1. *(cf. [19]) For a maximal monotone operator T, we say that its inverse T^{-1} is Lipschitz continuous at the origin (with modulus $a \geq 0$) if there is a unique solution $\bar{z}$ to $z = T^{-1}(0)$, and for some $\tau > 0$ we have*

$$\|z - \bar{z}\| \leq a\|w\|, \ \textit{where } z \in T^{-1}(w) \textit{ and } \|w\| \leq \tau.$$

We state the following lemma which will be needed later in the derivation of the PPA method for solving (P).

Lemma 2.1. *Let Y be an $n \times n$ symmetric matrix with eigenvalue decomposition $Y = PDP^T$ with $D = \mathrm{diag}(d)$. We assume that $d_1 \geq \cdots \geq d_r > 0 \geq d_{r+1} \cdots \geq d_n$. Let $\gamma > 0$ be given. For the two scalar functions $\phi_\gamma^+(x) := (\sqrt{x^2 + 4\gamma} + x)/2$ and $\phi_\gamma^-(x) := (\sqrt{x^2 + 4\gamma} - x)/2$ for all $x \in \Re$, we define their matrix counterparts:*

$$Y_1 = \phi_\gamma^+(Y) := P\mathrm{diag}(\phi_\gamma^+(d))P^T \quad \text{and} \quad Y_2 = \phi_\gamma^-(Y) := P\mathrm{diag}(\phi_\gamma^-(d))P^T. \tag{9}$$

Then
(a) The following decomposition holds: $Y = Y_1 - Y_2$, where $Y_1, Y_2 \succ 0$, and $Y_1 Y_2 = \gamma I$.
(b) ϕ_γ^+ is continuously differentiable everywhere in $\mathcal{S}^n$ and its derivative $(\phi_\gamma^+)'(Y)[H]$ at Y for any $H \in \mathcal{S}^n$ is given by

$$(\phi_\gamma^+)'(Y)[H] \ = \ P(\Omega \circ (P^T H P))P^T,$$

where $\Omega \in \mathcal{S}^n$ is defined by

$$\Omega_{ij} \ = \ \frac{\phi_\gamma^+(d_i) + \phi_\gamma^+(d_j)}{\sqrt{d_i^2 + 4\gamma} + \sqrt{d_j^2 + 4\gamma}}, \quad i, j = 1, \ldots, n.$$

(c) $(\phi_\gamma^+)'(Y)[Y_1 + Y_2] = \phi_\gamma^+(Y)$.

Proof. (a) It is easy to verify that the decomposition holds. (b) The result follows from [3, Ch. V.3.3] and the fact that

$$\frac{\phi_\gamma^+(d_i) - \phi_\gamma^+(d_j)}{d_i - d_j} \;=\; \frac{\phi_\gamma^+(d_i) + \phi_\gamma^+(d_j)}{\sqrt{d_i^2 + 4\gamma} + \sqrt{d_j^2 + 4\gamma}}, \quad d_i \neq d_j.$$

(c) We have $(\phi_\gamma^+)'(Y)[Y_1 + Y_2] \;=\; P\Big(\Omega \circ \mathrm{diag}(\phi_\gamma^+(d) + \phi_\gamma^-(d))\Big)P^T \;=\; P\mathrm{diag}(\phi_\gamma^+(d))P^T$, and the required result follows. □

3 The primal proximal point algorithm

Define the feasible sets of (P) and (D), respectively, by

$$\mathcal{F}_P = \{X \in \mathcal{S}^n \,:\, \mathcal{A}(X) = b, X \succeq 0\},\ \mathcal{F}_D = \{(y, Z) \in \mathcal{R}^m \times \mathcal{S}^n \,:\, Z + \mathcal{A}^T y = C, Z \succeq 0\}.$$

Throughout this paper, we assume that the following conditions for (P) and (D) hold.

Assumption 3.1. *Problem (P) satisfies the condition*

$$\exists\, X_0 \in \mathcal{S}^n_{++} \text{ such that } \mathcal{A}(X_0) = b. \tag{10}$$

Assumption 3.2. *Problem (D) satisfies the condition*

$$\exists\, (y_0, Z_0) \in \mathcal{R}^m \times \mathcal{S}^n_{++} \text{ such that } Z_0 + \mathcal{A}^T y_0 = C. \tag{11}$$

Under the above assumptions, problem (P) has a unique optimal solution, denoted by $\overline{X}$ and problem (D) has a unique optimal solution, denoted by $(\overline{y}, \overline{Z})$. In addition, the following Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient for the optimality of (P) and (D):

$$\begin{array}{rcl} \mathcal{A}(X) - b &=& 0, \\ Z + \mathcal{A}^T y - C &=& 0, \\ XZ &=& \mu I, \;\; X \succeq 0, \;\; Z \succeq 0. \end{array} \tag{12}$$

The last condition in (12) can be easily seen to be equivalent to the following condition

$$\phi_\gamma^+(X - \lambda Z) = X \quad \text{with } \gamma := \lambda\mu \tag{13}$$

for any given $\lambda > 0$, where ϕ_γ^+ is defined by (9) in Lemma 2.1. Recall that in [32] for the linear SDP case (where $\mu = 0$), the complementarity condition $XZ = 0$ with $X, Z \succeq 0$, is equivalent to $\Pi_+(X - \lambda Z) = X$, for any $\lambda > 0$, where $\Pi_+(\cdot)$ is the metric projector onto $\mathcal{S}^n_+$. One can see from (13) that when $\mu > 0$, the log-barrier term $-\mu \log \det X$ in (P) contributed to a smoothing term in the projector Π_+.

Lemma 3.1. *Given any $Y \in \mathcal{S}^n$ and $\lambda > 0$, we have*

$$\min_{Z \succ 0} \Big\{ \frac{1}{2\lambda}\|Y - Z\|^2 - \mu \log\det Z \Big\} = \frac{1}{2\lambda}\|\phi_\gamma^-(Y)\|^2 - \mu \log\det(\phi_\gamma^+(Y)), \tag{14}$$

where $\gamma = \lambda\mu$.

Proof. Note that the minimization problem in (14) is an unconstrained problem and the objective function is strictly convex and continuously differentiable. Thus any stationary point would be the unique minimizer of the problem. The stationary point, if it exists, is the solution of the following equation:

$$Y = Z - \gamma Z^{-1}. \tag{15}$$

By Lemma 2.1(a), we see that $Z_* := \phi_\gamma^+(Y)$ satisfies (15). Thus the optimization problem in (14) has a unique minimizer and the minimum objective function value is given by

$$\frac{1}{2\lambda}\|Y - Z_*\|^2 - \mu \log\det Z_* = \frac{1}{2\lambda}\|\phi_\gamma^-(Y)\|^2 - \mu \log\det(\phi_\gamma^+(Y)).$$

This completes the proof. □

Let $l(X; y) : \mathcal{S}^n \times \mathcal{R}^m \to \mathcal{R}$ be the ordinary Lagrangian function for (P) in extended form:

$$l(X; y) = \begin{cases} \langle C,\, X\rangle - \mu \log\det X + \langle y,\, b - \mathcal{A}(X)\rangle & \text{if } X \in \mathcal{S}^n_+, \\ \infty & \text{otherwise.} \end{cases} \tag{16}$$

The essential objective function in (P) is given by

$$f(X) = \max_{y \in \mathcal{R}^m} l(X; y) = \begin{cases} \langle C,\, X\rangle - \mu \log\det X & \text{if } X \in \mathcal{F}_P, \\ \infty & \text{otherwise.} \end{cases} \tag{17}$$

For later developments, we define the following maximal monotone operator associated with $l(X, y)$:

$$T_l(X, y) \;:=\; \{(U, v) \in \mathcal{S}^n \times \mathcal{R}^m \,|\, (U, -v) \in \partial l(X, y),\;\; (X, y) \in \mathcal{S}^n \times \mathcal{R}^m\}.$$

Let F_λ be the Moreau-Yosida regularization (see [15, 30]) of f in (17) associated with $\lambda > 0$, i.e.,

$$F_\lambda(X) = \min_{Y \in \mathcal{S}^n} \{f(Y) + \frac{1}{2\lambda}\|Y - X\|^2\} = \min_{Y \in \mathcal{S}^n_{++}} \{f(Y) + \frac{1}{2\lambda}\|Y - X\|^2\}. \tag{18}$$

From (17), we have

$$\begin{aligned} F_\lambda(X) &= \min_{Y \in \mathcal{S}^n_{++}} \sup_{y \in \mathcal{R}^m} \Big\{ l(Y; y) + \frac{1}{2\lambda}\|Y - X\|^2 \Big\} \\ &= \sup_{y \in \mathcal{R}^m} \min_{Y \in \mathcal{S}^n_{++}} \Big\{ l(Y; y) + \frac{1}{2\lambda}\|Y - X\|^2 \Big\} \;=\; \sup_{y \in \mathcal{R}^m} \Theta_\lambda(X, y)\,, \end{aligned} \tag{19}$$

where

$$\begin{aligned}\Theta_\lambda(X,y) &= \min_{Y\in\mathcal{S}^n_{++}} \Big\{ l(Y;y) + \frac{1}{2\lambda}\|Y-X\|^2 \Big\} \\ &= b^Ty + \min_{Y\in\mathcal{S}^n_{++}} \Big\{ \langle C - \mathcal{A}^Ty,\, Y\rangle - \mu\log\det Y + \frac{1}{2\lambda}\|Y-X\|^2 \Big\} \\ &= b^Ty - \frac{1}{2\lambda}\|W_\lambda(X,y)\|^2 + \frac{1}{2\lambda}\|X\|^2 + \min_{Y\in\mathcal{S}^n_{++}} \Big\{ \frac{1}{2\lambda}\|Y-W_\lambda(X,y)\|^2 - \mu\log\det Y \Big\}.\end{aligned}$$

Here $W_\lambda(X,y) := X - \lambda(C - \mathcal{A}^Ty)$. Note that the interchange of min and sup in (19) follows from [18, Theorem 37.3]. By Lemma 3.1, the minimum objective value in the above minimization problem is attained at $Y_* = \phi^+_\gamma(W_\lambda(X,y))$. Thus we have

$$\Theta_\lambda(X,y) = b^Ty + \frac{1}{2\lambda}\|X\|^2 - \frac{1}{2\lambda}\|\phi^+_\gamma(W_\lambda(X,y))\|^2 - \mu\log\det\phi^+_\gamma(W_\lambda(X,y)) + n\mu. \quad (20)$$

Note that for a given X, the function $\Theta_\lambda(X,\cdot)$ is analytic, cf. [26]. Its first and second order derivatives with respect to y can be computed as in the following lemma.

Lemma 3.2. *For any $y \in \mathcal{R}^m$ and $X \succ 0$, we have*

$$\nabla_y\Theta_\lambda(X,y) = b - \mathcal{A}\phi^+_\gamma(W_\lambda(X,y)) \quad (21)$$

$$\nabla^2_{yy}\Theta_\lambda(X,y) = -\lambda\mathcal{A}(\phi^+_\gamma)'(W_\lambda(X,y))\mathcal{A}^T \quad (22)$$

Proof. To simplify notation, we use W to denote $W_\lambda(X,y)$ in this proof. To prove (21), note that

$$\begin{aligned}\nabla_y\Theta_\lambda(X,y) &= b - \mathcal{A}(\phi^+_\gamma)'(W)[\phi^+_\gamma(W)] - \lambda\mu\mathcal{A}(\phi^+_\gamma)'(W)[(\phi^+_\gamma(W))^{-1}] \\ &= b - \mathcal{A}(\phi^+_\gamma)'(W)[\phi^+_\gamma(W) + \phi^-_\gamma(W)].\end{aligned}$$

By Lemma 2.1(c), the required result follows. From (21), the result in (22) follows readily. □

Let $y_\lambda(X)$ be such that

$$y_\lambda(X) \in \arg\sup_{y\in\mathcal{R}^m} \Theta_\lambda(X,y).$$

Then we know that $\phi^+_\gamma(W_\lambda(X,y_\lambda(X)))$ is the unique optimal solution to (18). Consequently, we have that $F_\lambda(X) = \Theta_\lambda(X,y_\lambda(X))$ and

$$\nabla F_\lambda(X) = \frac{1}{\lambda}\Big(X - \phi^+_\gamma(W_\lambda(X,y_\lambda(X)))\Big) = C - \mathcal{A}^Ty - \frac{1}{\lambda}\phi^-_\gamma(W_\lambda(X,y_\lambda(X))). \quad (23)$$

Given $X^0 \in \mathcal{S}^n_{++}$, the exact PPA for solving (P), is given by

$$X^{k+1} \;=\; (I + \lambda_k T_f)^{-1}(X^k) = \arg\min_{X \in \mathcal{S}^n_{++}} \Big\{ f(X) + \frac{1}{2\lambda_k}\|X - X^k\|^2 \Big\}, \tag{24}$$

where $T_f = \partial f$. It can be shown [21, Theorem 2.26] that

$$X^{k+1} = X^k - \lambda_k \nabla F_{\lambda_k}(X^k) = \phi^+_{\gamma_k}(W_\lambda(X^k, y_{\lambda_k}(X^k))), \tag{25}$$

where $\gamma_k = \lambda_k \mu$.

The exact PPA outlined in (24) is impractical for computational purpose. Hence we consider an inexact PPA for solving (P), which has the following template.

Algorithm 1: The Primal PPA. Given a tolerance $\varepsilon > 0$. Input $X^0 \in \mathcal{S}^n_{++}$ and $\lambda_0 > 0$. Set $k := 0$. Iterate:

Step 1. Find an approximate maximizer

$$y^{k+1} \approx \arg\sup_{y \in \mathcal{R}^m} \Big\{ \theta_k(y) := \Theta_{\lambda_k}(X^k, y) \Big\}, \tag{26}$$

where $\Theta_{\lambda_k}(X^k, y)$ is defined as in (20).

Step 2. Compute

$$X^{k+1} \;=\; \phi^+_{\gamma_k}(W_{\lambda_k}(X^k, y^{k+1})), \quad Z^{k+1} \;=\; \frac{1}{\lambda_k}\phi^-_{\gamma_k}(W_{\lambda_k}(X^k, y^{k+1})). \tag{27}$$

Step 3. If $\|(X^k - X^{k+1})/\lambda_k\| \le \epsilon$; stop; else; $\lambda_k = 2\lambda_k$; end.

Remark 3.1. *Note that* $b - \mathcal{A}(X^{k+1}) = b - \mathcal{A}\phi^+_{\gamma_k}(W_{\lambda_k}(X^k, y^{k+1})) = \nabla_y \Theta_{\lambda_k}(X^k, y^{k+1}) \approx 0$.

Remark 3.2. *Observe that the function $\Theta_\lambda(X, y)$ is twice continuously differentiable (actually, analytic) in y. In contrast, its counterpart $L_\sigma(y, X)$ for a linear SDP in [32] fails to be twice continuously differentiable in y and only the Clarke's generalized Jacobian of $\nabla_y L_\sigma(y, X)$ (i.e., $\partial \nabla_y L_\sigma(y, X)$) can be obtained. This difference can be attributed to the term $-\mu \log\det X$ in problem (P). In other words, $-\mu \log\det X$ works as a smoothing term that turns $L_\sigma(y, X)$ (which is not twice continuously differentiable) into an analytic function in y. This idea is different from the traditional smoothing technique of using a smoothing function on the KKT conditions since the latter is not motivated by adding a smoothing term to an objective function. Our derivation of $\Theta_\lambda(y, X)$ shows that the smoothing technique of using a squared smoothing function $\phi^+_\gamma(x) = (\sqrt{x^2 + 4\gamma} + x)/2$ can indeed be derived by adding the log-barrier term to the objective function.*

The advantage of viewing the smoothing technique from the perspective of adding a log-barrier term is that the error between the minimum objective function values of the perturbed problem and the original problem can be estimated. In the traditional smoothing technique for the KKT conditions, there is no obvious mean to estimate the error in the objective function value of the solution computed from the smoothed KKT conditions from the true minimum objective function value. We believe the connection we discovered here could be useful for the error analysis of the smoothing technique applied to the KKT conditions.

For the sake of subsequent convergence analysis, we present the following proposition.

Proposition 3.1. *Suppose that (P) satisfies (10). Let $\overline{X} \in S^n_{++}$ be the unique optimal solution to (P), i.e., $\overline{X} = T_f^{-1}(0)$. Then T_f^{-1} is Lipschitz continuous at the origin.*

Proof. From [20, Prop. 3], it suffices to show that the following quadratic growth condition holds at $\overline{X}$ for some positive constant α:

$$f(X) \geq f(\overline{X}) + \alpha \|X - \overline{X}\|^2 \quad \forall\ X \in \mathcal{N} \text{ such that } X \in \mathcal{F}_P \tag{28}$$

where $\mathcal{N}$ is a neighborhood of $\overline{X}$ in S^n_{++}. From [4, Theorem 3.137], to prove (28), it suffices to show the second order sufficient condition for (P) holds.

Now for $X \in \mathcal{S}^n_{++}$, we have

$$\langle \Delta X, \nabla^2_{XX} l(X; y)(\Delta X)\rangle \ = \ \mu \langle X^{-1}\Delta X X^{-1}, \Delta X\rangle \ \geq \ \mu \lambda^{-2}_{\max}(X)\|\Delta X\|^2, \ \forall\, \Delta X \in \mathcal{S}^n,$$

where $\lambda_{\max}(X)$ is the maximal eigenvalue of X, this is equivalent to

$$\langle \Delta X, \nabla^2_{XX} l(X; y)(\Delta X)\rangle > 0, \ \forall\, \Delta X \in \mathcal{S}^n \setminus \{0\}. \tag{29}$$

Certainly, (29) implies the second order sufficient condition for Problem (P). □

We can also prove in parallel that the maximal monotone operator T_l is Lipschitz continuous at the origin.

4 The Newton-CG method for inner problems

In the algorithm framework proposed in Section 3, we have to compute $y^{k+1} \approx \arg\sup \{\theta_k(y) : y \in \mathcal{R}^m\}$. In this paper, we will introduce the Newton-CG method to achieve this goal.

Algorithm 2: The Newton-CG Method.

Step 0. Given $\mu \in (0, \frac{1}{2}),\ \tau_1, \tau_2 \in (0,1),$ and $\delta \in (0,1)$, choose $y^0 \in \mathcal{R}^m$.

Step 1. For $j = 0, 1, 2, \ldots,$

Step 1.1. Apply the PCG method to find an approximate solution d^j to

$$(\nabla^2_{yy}\theta_k(y^j) - \epsilon_j I)d = -\nabla_y \theta_k(y^j), \tag{30}$$

where $\epsilon_j := \tau_1 \min\{\tau_2, \|\nabla_y \theta_k(y^j)\|\}$.

Step 1.2. Set $\alpha_j = \delta^{m_j}$, where m_j is the first nonnegative integer m for which

$$\theta_k(y^j + \delta^m d^j) \geq \theta_k(y^j) + \mu\delta^m \langle \nabla_y \theta_k(y^j), d^j \rangle.$$

Step 1.3. Set $y^{j+1} = y^j + \alpha_j d^j$.

From (22) and the positive definiteness property of $\phi'_+(W(y;X))$ (for some properties of the projection operator one may refer to [13]), we have that $-\nabla^2_{yy}\theta_k(y^j)$ is always positive definite, then $-\nabla^2_{yy}\theta_k(y^j) + \epsilon_j I$ is positive definite as long as $\nabla_y \theta_k(y^j) \neq 0$. So we can always apply the PCG method to (30). Of course, the direction d^j generated from (30) is always an ascent direction. With respect to the analysis of the global convergence and local quadratic convergence rate of the above algorithm, we will not present the details, and one may refer to Section 3.3 of [32] since it is very similar to the semismooth Newton-CG algorithm used in that paper. The difference lies in that d^j obtained from (30) in this paper is an approximate Newton direction; in contrast, d^j obtained from (61) in [32] is a semismooth Newton direction.

5 Convergence analysis

Global convergence and the local convergence rate of our Newton-CG PPA method to problems (P) and (D) can directly be derived from Rockafellar's papers [19, 20] without much difficulty. For the sake of completeness, we shall only state the results below.

Since we cannot solve the inner problems exactly, we will use the following stopping criteria considered by Rockafellar [19, 20] for terminating Algorithm 2:

$$(A)\ \sup \theta_k(y) - \theta_k(y^{k+1}) \leq \epsilon_k^2/2\lambda_k,\quad \epsilon_k \geq 0, \sum_{k=0}^{\infty} \epsilon_k < \infty;$$

$$(B)\ \sup \theta_k(y) - \theta_k(y^{k+1}) \leq \delta_k^2/2\lambda_k \|X^{k+1} - X^k\|^2,\quad \delta_k \geq 0, \sum_{k=0}^{\infty} \delta_k < \infty;$$

$$(B')\ \|\nabla_y \theta_k(y^{k+1})\| \leq \delta'_k/\lambda_k \|X^{k+1} - X^k\|,\quad 0 \leq \delta'_k \to 0.$$

In view of Proposition 3.1, we can directly obtain from [19, 20] the following convergence results.

Theorem 5.1. *Let Algorithm 1 be executed with stopping criterion (A). If (D) satisfies condition (11), then the generated sequence* $\{X^k\} \subset \mathcal{S}^n_{++}$ *is bounded and* $\{X^k\}$ *converges to* $\overline{X}$*, where* $\overline{X}$ *is the unique optimal solution to (P), and* $\{y^k\}$ *is asymptotically maximizing for (D) with min(P)=sup(D).*

If $\{X^k\}$ *is bounded and (P) satisfies condition (10), then the sequence* $\{y^k\}$ *is also bounded, and the accumulation point of the sequence* $\{y^k\}$ *is the unique optimal solution to (D).*

Theorem 5.2. *Let Algorithm 1 be executed with stopping criteria (A) and (B). Assume that (D) satisfies condition (11) and (P) satisfies condition (10). Then the generated sequence* $\{X^k\} \subset \mathcal{S}^n_{++}$ *is bounded and* $\{X^k\}$ *converges to the unique solution* $\overline{X}$ *to (P) with min(P)=sup(D), and*

$$\|X^{k+1} - \overline{X}\| \leq \theta_k \|X^k - \overline{X}\|, \text{ for all } k \text{ sufficiently large,}$$

where

$$\theta_k = [a_f(a_f^2 + \sigma_k^2)^{-1/2} + \delta_k](1 - \delta_k)^{-1} \to \theta_\infty = a_f(a_f^2 + \sigma_\infty^2)^{-1/2} < 1, \sigma_k \to \sigma_\infty,$$

and a_f *is a Lipschitz constant of* T_f^{-1} *at the origin. The conclusions of Theorem 5.1 about* $\{y^k\}$ *are valid.*

Moreover, if the stopping criterion (B′) is also used, then in addition to the above conclusions the sequence $\{y^k\} \to \overline{y}$*, where* $\overline{y}$ *is the unique optimal solution to (D), and one has*

$$\|y^{k+1} - \overline{y}\| \leq \theta'_k \|X^{k+1} - X^k\|, \text{ for all } k \text{ sufficiently large,}$$

where

$$\theta'_k = a_l(1 + \delta'_k)/\sigma_k \to \delta_\infty = a_l/\sigma_\infty,$$

and a_l *is a Lipschitz constant of* T_l^{-1} *at the origin.*

Remark 5.1. *In Algorithm 1 we can also add the term* $-\frac{1}{2\lambda_k}\|y - y^k\|^2$ *to* $\theta_k(y)$*. Actually, in our* MATLAB *code, one can optionally add this term. This actually corresponds to the PPA of multipliers considered in [20, Section 5]. Convergence analysis for this improvement can be conducted in a parallel way as for Algorithm 1.*

Note that in the stopping criteria (A) and (B), $\sup \theta_k(y)$ *is an unknown value. Since* $\hat{\theta}_k(y) := \theta_k(y) - \frac{1}{2\lambda_k}\|y - y^k\|^2$ *is a strongly concave function with modulus* $\frac{1}{\lambda_k}$*, then one has the estimation*

$$\sup \hat{\theta}_k(y) - \hat{\theta}_k(y^{k+1}) \leq \frac{1}{2\lambda_k}\|\nabla_y \hat{\theta}_k(y^{k+1})\|^2,$$

thus criteria (A) and (B) can be practically modified as follows:

$$\|\nabla_y \hat{\theta}_k(y^{k+1})\| \le \epsilon_k, \quad \epsilon_k \ge 0, \sum_{k=0}^{\infty} \epsilon_k < \infty;$$

$$\|\nabla_y \hat{\theta}_k(y^{k+1})\| \le \delta_k \|X^{k+1} - X^k\|, \quad \delta_k \ge 0, \sum_{k=0}^{\infty} \delta_k < \infty.$$

6 Numerical experiments

In this section, we present some numerical results to demonstrate the performance of our PPA on (4) and (5), for Gaussian graphical models with both synthetic and real data. We implemented the PPA in MATLAB. All runs are performed on an Intel Xeon 3.20GHz PC with 4GB memory, running Linux and MATLAB (Version 7.6).

We measure the infeasibilities and optimality for the primal and dual problems (P) and (D) as follows:

$$R_D = \frac{\|C - \mathcal{A}^T y - Z\|}{1 + \|C\|}, \quad R_P = \frac{\|b - \mathcal{A}(X)\|}{1 + \|b\|}, \quad R_G = \frac{|\texttt{pobj} - \texttt{dobj}|}{1 + |\texttt{pobj}| + |\texttt{dobj}|}, \tag{31}$$

where $\texttt{pobj} = \langle C,\, X\rangle - \mu \log\det X$ and $\texttt{dobj} = b^T y + \mu \log\det Z + n\mu(1 - \log\mu)$. The above measures are similar to those adopted in [32]. In our numerical experiments, we stop the PPA when

$$\max\{R_D, R_P\} \le \texttt{Tol}, \tag{32}$$

where `Tol` is a pre-specified accuracy tolerance. Note that the third equation $XZ = \mu I$ in (12) holds up to machine precision because of the way we define Z in (27) in the PPA. Unless otherwise specified, we set $\texttt{Tol} = 10^{-6}$ as the default. We choose the initial iterate $X^0 = I$, and $\lambda_0 = 1$.

We should note that in the PPA, computing the full eigenvalue decomposition of the matrix $W_{\lambda_k}(X^k, y)$ to evaluate the function $\Theta_{\lambda_k}(X^k, y)$ in (26) may constitute a major part of the overall computation. Thus it is essential for us to use an eigenvalue decomposition routine that is as efficient as possible. In our implementation, we use the LAPACK routine `dsyevd.f` (based on a divide-and-conquer strategy) to compute the full eigenvalue decomposition of a symmetric matrix. On our machine, it is about 7 to 10 times faster than MATLAB's `eig` routine when n is larger than 500. In the ANS method of [12, 11], having an efficient eigenvalue decomposition routine is even more crucial. Thus in our experiments, we also use the faster eigenvalue routine for the ANS method.

We focus our numerical experiments on the problems (4) and (5). The problem (5) is not expressed in the standard form given in (2), but it can easily be expressed as such by introducing additional constraints and variables. To be precise, the standard form

reformulation of (5) is given as follows:

$$\begin{array}{ll} \min & \langle C,\, X\rangle - \mu \log\det X + \rho^T x^+ + \rho^T x^- - \nu \log x^+ - \nu \log x^- \\ \text{s.t.} & X_{ij} = 0 \quad \forall\, (i,j) \in \Omega \\ & X_{ij} - x^+_{ij} + x^-_{ij} = 0 \quad \forall\, (i,j) \notin \Omega \\ & X \succeq 0, \quad x^+, x^- \geq 0, \end{array} \tag{33}$$

where we set $\nu = 10^{-16}$ (Originally, $\nu = 0$, however, for the convenience of the theory, we set $\nu = 10^{-16}$ in practical computation), $\rho, x^+, x^- \in \mathcal{R}^{m_2}$, $m_2 = m - m_1$, $m = \frac{n(n+1)}{2}$, $m_1 = \frac{1}{2}|\Omega|$ ($|\Omega|$ denotes the number of the elements in the index set Ω).

We should emphasize that our algorithm is sensitive to the scaling of the data, especially for problem (5). Thus in our implementation, we first scale the data by setting $A_k \leftarrow A_k/\|A_k\|$, $C \leftarrow C/\|C\|$ and $b \leftarrow b/\|b\|$.

In this paper, we mainly compare the performance of our PPA with the ANS method in [11, 12], whose MATLAB codes are available at `http://www.math.sfu.ca/~zhaosong/`. The reason for comparing only with the ANS method is because it is currently the most advanced first order method developed for solving the covariance selection problems (4) and (5). In the ANS method, the convergence criterion is controlled by two parameters ϵ_o, ϵ_c, which stand for the errors in the objective value and primal infeasibility, respectively.

As mentioned in [28, 29], we may evaluate the performance of an estimator $\widehat{\Sigma}$ of the true covariance matrix Σ by a normalized L_2-loss function which is defined as follows:

$$L_2^{\rm loss} = \|\Sigma^{-1}\widehat{\Sigma} - I\|_F/n.$$

Thus in our numerical experiments, we also report the above value when it is possible to do so.

6.1 Some acceleration techniques

6.1.1 A diagonal preconditioner for (30)

To achieve faster convergence for the CG method to solve (30), one may apply a proper preconditioner to the linear system. But a suitable balance between having an effective preconditioner and additional computational cost must be determined. In our implementation, we devise an easy-to-compute diagonal preconditioner by using an idea first developed in [9].

Let M denotes the coefficient matrix in the left-hand side of (30), which has the form:

$$M := -\lambda \mathbf{A}\mathbf{T}\mathbf{A}^T - \epsilon I,$$

where $\mathbf{A}$ and $\mathbf{T}$ denote the matrix representation of the linear map $\mathcal{A}$ and $\mathcal{T}$ with respect to the standard bases in $\mathcal{S}^n$ and $\mathcal{R}^m$, respectively. Here $\mathcal{T} : \mathcal{S}^n \to \mathcal{S}^n$ is the linear operator defined by $\mathcal{T}(X) = P(\Omega \circ (P^T X P))P^T$.

Recall that the standard basis in $\mathcal{S}^n$ is given by $\{E_{ij} := \alpha_{ij}(e_i e_j^T + e_j e_i^T) : 1 \le i \le j \le n\}$, where e_i is the ith unit vector in $\mathcal{R}^n$, and $\alpha_{ij} = 1/\sqrt{2}$ if $i \neq j$ and $\alpha_{ij} = 1/2$ otherwise. Then the diagonal element of $\mathbf{T}$ with respect to the basis element E_{ij} is given by

$$\mathbf{T}_{(ij),(ij)} = \langle P^T E_{ij} P, \Omega \circ (P^T E_{ij} P) \rangle = \begin{cases} ((P \circ P)^T \Omega (P \circ P))_{ij} + \langle v^{(ij)}, \Omega v^{(ij)} \rangle & \text{if } i \neq j \\ ((P \circ P)^T \Omega (P \circ P))_{ij} & \text{otherwise} \end{cases} \tag{34}$$

where $v^{(ij)} = P_i \circ P_j$ and P_i, P_j are the ith and jth columns of P, respectively. From (34) we can see that to compute all the diagonal entries of $\mathbf{T}$, the computing cost of $O(n^4)$ flops is needed. Fortunately, the first term

$$\mathbf{d}_{(ij)} := ((P \circ P)^T \Omega (P \circ P))_{ij} \tag{35}$$

in the right-hand side of (34) is typically a very good approximation of $\mathbf{T}_{(ij),(ij)}$. More importantly, computing all the elements $\mathbf{d}_{(ij)}$, for $1 \le i \le j \le n$, needs only $O(n^3)$ flops since only the matrix product $(P \circ P)^T \Omega (P \circ P)$ is involved. We propose the following diagonal preconditioner for M:

$$M_D := -\lambda \mathrm{diag}(\mathbf{A}\mathrm{diag}(\mathbf{d})\mathbf{A}^T) - \epsilon I. \tag{36}$$

6.1.2 An alternating Newton PPA

In this subsection, we introduce an alternating Newton PPA to accelerate the convergence of the primal proximal point algorithm for solving the problem (P).

For the PPA, we know that the outer iteration is actually a gradient method, i.e.,

$$X^{k+1} = X^k - \lambda_k \nabla F_{\lambda_k}(X^k) = \phi_{\gamma_k}^+(W_{\lambda_k}(X^k, y_{\lambda_k}(X^k))). \tag{37}$$

To improve the convergence rate of the outer iteration, we may use Newton's method instead. Since

$$\nabla F_\lambda(X) = \frac{1}{\lambda}\Big(X - \phi_\gamma^+(W_\lambda(X, y_\lambda(X)))\Big),$$

we have that

$$\nabla^2 F_\lambda(X)[H] = \frac{1}{\lambda}[H - (\phi_\gamma^+)'(W)(H + \lambda \mathcal{A}^T y_\gamma'(X; H))], \ \forall\, H \in \mathcal{S}^n. \tag{38}$$

Here we used W to denote $W_\lambda(X, y_\lambda(X))$ to simplify the notation. Thus, the Newton direction H^k is the solution to the following linear system of equations:

$$\nabla^2 F_\lambda(X^k)[H^k] = -\nabla F_\lambda(X^k). \tag{39}$$

Once H^k has been computed, the new iteration is updated by (if the iteration X^k is sufficiently close to the optimal solution):

$$X^{k+1} = X^k + H^k. \tag{40}$$

In practical computation, we adopt the update based on Newton's method in (40) (without line search) when the relative primal infeasibility and dual feasibility are both less than 10^{-2}; otherwise, we use the update in (37). We call the PPA with its outer iteration updated possibly by Newton's update (40) the alternating Newton PPA and we abbreviate it by ANPPA.

Next we discuss how the Newton system (39) can be solved by the CG method. To apply the CG method, all we need is to be able to evaluate $\nabla^2 F_\lambda(X^k)[H]$ given any $H \in \mathcal{S}^n$. Now we discuss in detail how this can be done. Observe that since

$$0 = \nabla_y \Theta_\lambda(X, y_\lambda(X)) = b - \mathcal{A}\phi_\gamma^+(W_\lambda(X, y_\lambda(X)),$$

it follows that

$$0 = -\mathcal{A}(\phi_\gamma^+)'(W)[H + \lambda \mathcal{A}^T y_\gamma'(X; H)],$$

and

$$\lambda \mathcal{A}(\phi_\gamma^+)'(W)\mathcal{A}^T y_\gamma'(X; H) = -\mathcal{A}(\phi_\gamma^+)'(W)[H]. \tag{41}$$

Thus from the linear system (41), we can compute $y_\gamma'(X; H)$ given any X and H. By substituting it into (38), we can evaluate the expression, $\nabla^2 F_\lambda(X)[H]$ for any given X and H.

To summarize, given X and H, we can evaluate $\nabla^2 F_\lambda(X)[H]$ by solving the linear system (41) for $y_\gamma'(X; H)$. Observe that the linear system (41) has exactly the same form as the linear system (30), thus whatever techniques we have developed to solve (30), we can use them to solve (41). Note that we terminate the CG method for solving (39) when the relative residual norm is less than 5×10^{-2}.

6.2 Synthetic experiments I

All instances used in this section were randomly generated in a similar manner as described in d'Aspremont et al. [8]. Indeed, we generate a random sparse positive definite matrix $\Sigma^{-1} \in \mathcal{S}_{++}^n$ with a density of about 10% non-zero entries as follows. First we generate an $n \times n$ random sparse matrix $\mathtt{U}$ with non-zero entries set to ± 1. Then set

$$\mathtt{A} = \mathtt{U}' * \mathtt{U}; \quad \mathtt{d} = \mathtt{diag}(\mathtt{A}); \quad \mathtt{A} = \mathtt{max}(\mathtt{min}(\mathtt{A} - \mathtt{diag}(\mathtt{d}), 1), -1);$$

$$\mathtt{B} = \mathtt{A} + \mathtt{diag}(1 + \mathtt{d});$$

$$\Sigma^{-1} = \mathtt{B} + \mathtt{max}(-1.2 * \mathtt{min}(\mathtt{eig}(\mathtt{B})), 0.001) * \mathtt{I};$$

The sample covariance matrix S for (4) is generated in a similar manner as in [8], [11] via the following script:

$$\mathtt{E} = 2 * \mathtt{rand}(\mathtt{n}) - 1; \quad \mathtt{E} = 0.5 * (\mathtt{E} + \mathtt{E}');$$

$$\mathtt{S} = \Sigma + 0.15 * (\|\mathtt{Sigma}\|_{\mathtt{F}} / \|\mathtt{E}\|_{\mathtt{F}}) * \mathtt{E};$$

$$\mathtt{S} = \mathtt{S} + \mathtt{max}(-\mathtt{min}(\mathtt{eig}(\mathtt{S})), 0.001) * \mathtt{I};$$

The set Ω is generated as in the MATLAB codes developed by Lu for the paper [11], specifically,

$$\Omega = \{(i,j) : (\Sigma^{-1})_{ij} = 0,\ |i-j| \geq 5\}.$$

In this synthetic experiment, we apply the alternating Newton primal proximal point algorithm (ANPPA) to the problem (4). The performance of the ANPPA is presented in Table 1. For each instance in the table, we report the matrix dimension (n); the number of linear constraints (m); the number of outer iterations (it), the total number of subproblems ($itsub$) solved by the ANPPA, and the average number of PCG steps (pcg) taken to solve each linear system in (30); the primal (`pobj`) and dual (`dobj`) objective values; the primal (R_P) and dual (R_D) infeasibilities, the relative gap (R_G), and $L_2^{\rm loss}$; the time (in seconds) taken. We may observe from the table that the ANPPA can very efficiently solve the problem (4) with synthetic data.

Table 1: Performance of the ANPPA on (4) with synthetic data (I).

problem	m | n	it/$itsub$/pcg	`pobj`	`dobj`	$R_P/R_D/R_G/L_2^{\rm loss}$	time
rand-500	112172 | 500	12| 23| 9.9	-1.85356498 2	-1.85356176 2	4.1-7| 2.9-7| 8.7-7| 1.8-2	43.1
rand-1000	441294 | 1000	11| 22| 10.0	-1.22155120 2	-1.22155034 2	8.0-8| 4.7-8| 3.5-7| 6.7-2	247.0
rand-1500	979620 | 1500	11| 22| 8.9	-4.32799516 2	-4.32799143 2	2.2-7| 1.4-7| 4.3-7| 6.8-2	698.0
rand-2000	1719589 | 2000	12| 23| 8.7	-1.34583154 3	-1.34583004 3	8.1-7| 3.5-7| 5.6-7| 4.3-2	1581.4

In Table 2, we compare the performance of our ANPPA and the ANS method on the first three instances reported in Table 1. For the ANPPA, `Tol` in (32) is set to $10^{-6}, 10^{-7}$ and 10^{-8}; for ANS, ϵ_o is set to $10^{-1}, 10^{-2}$ and 10^{-3}, and ϵ_c is set to 10^{-4}, so that the gap ($= |\texttt{pobj} - \texttt{dobj}|$) can fall below $10^{-1}, 10^{-2}$ and 10^{-3}, respectively. For each instance in the table, we give the matrix dimension (n) and the number of linear constraints (m); the gaps achieved, and the times taken (in seconds). From Table 2, we can see that both methods are able to solve all instances within a reasonable amount of time. However, the ANPPA consistently outperforms the ANS method by a factor which ranges from 3 to 15.

Table 2: Comparison of the ANPPA and the ANS method on (4) with synthetic data (I).

problem	m | n	tolerance		iteration		gap		time	
		PPA (`Tol`)	ANS (ϵ_o, ϵ_c)	PPA	ANS	PPA	ANS	PPA	ANS
rand-500	112172 | 500	3×10^{-6}	$(10^{-1}, 10^{-4})$	23	534	3.22-4	9.70-2	43.1	133.8
		3×10^{-7}	$(10^{-2}, 10^{-4})$	27	1255	1.18-6	9.94-3	55.2	285.9
		3×10^{-8}	$(10^{-3}, 10^{-4})$	27	2945	1.18-6	9.94-4	55.2	646.5
rand-1000	441294 | 1000	3×10^{-6}	$(10^{-1}, 10^{-4})$	22	826	8.52-5	9.66-2	247.0	1137.5
		3×10^{-7}	$(10^{-2}, 10^{-4})$	22	1916	8.52-5	9.94-3	247.0	2407.9
		3×10^{-8}	$(10^{-3}, 10^{-4})$	25	3924	2.39-6	9.94-4	304.8	4757.5
rand-1500	979620 | 1500	3×10^{-6}	$(10^{-1}, 10^{-4})$	22	778	3.73-4	9.94-2	698.0	3303.7
		3×10^{-7}	$(10^{-2}, 10^{-4})$	22	1741	3.73-4	9.94-3	698.0	6706.3
		3×10^{-8}	$(10^{-3}, 10^{-4})$	25	3452	1.17-5	9.94-4	857.0	12766.2

6.3 Synthetic experiments II

We note that the procedure used in [8] to generate the data matrix S for (4) is not in line with the standard practice in statistics. But since the covariance selection problem is a problem in statistics, we prefer to generate the data matrix S according to the standard practice; see for example, [28, 29]. Thus in this sub-section, the true covariance matrices Σ and the index sets Ω are generated exactly the same way as in the previous sub-section. But the sample covariance matrices S are generated differently. For each test problem, we sample $2n$ instances from the multivariate Gaussian distribution $N(0, \Sigma)$ to generate a sample covariance matrix S.

In the first synthetic experiment, we apply the ANPPA to the problem (4). The performance of the ANPPA is presented in Table 3. Again, the ANPPA can very efficiently solve the problem (4) with S generated from $2n$ samples of the Gaussian distribution $N(0, \Sigma)$. Comparing with Table 1, it appears that the log-det problems in Table 3 are harder to solve when n is large.

In Figure 1, we show that the ANPPA can also obtain very accurate solution for the instance `rand-500` reported in Table 3 without incurring substantial amount of additional computing time. As can be seen from the figure, the time taken only grows almost linearly when the required accuracy is geometrically reduced.

Table 3: Performance of the ANPPA on (4) with synthetic data (II).

problem	m | n	*it/itsub/pcg*	`pobj`	`dobj`	$R_P/R_D/R_G/L_2^{\text{loss}}$	time
rand-500	112172 | 500	13| 27| 13.2	-3.13591727 2	-3.13591672 2	2.6-8| 3.8-8| 8.8-8| 1.7-2	61.4
rand-1000	441294 | 1000	13| 29| 18.9	-9.74364421 2	-9.74360450 2	3.1-7| 6.0-7| 2.0-6| 2.0-2	468.8
rand-1500	979620 | 1500	13| 29| 15.8	-1.91034252 3	-1.91033842 3	7.1-7| 3.5-7| 1.1-6| 1.8-2	1384.5
rand-2000	1719589 | 2000	15| 34| 15.9	-3.00395927 3	-3.00395919 3	1.7-8| 5.0-9| 1.4-8| 1.5-2	3696.2

In Table 4, we compare the performance of our ANPPA and the ANS method on the first three instances reported in Table 3. For the ANPPA, `Tol` in (32) is set to $3 \times 10^{-6}, 3 \times 10^{-7}$ and 3×10^{-8}; for ANS, ϵ_o is set to $10^{-1}, 10^{-2}$ and 10^{-3}, and ϵ_c is set to 10^{-4}, so that the gap ($= |\texttt{pobj} - \texttt{dobj}|$) can fall below $10^{-1}, 10^{-2}$ and 10^{-3}, respectively. From Table 4, we can see that the ANPPA consistently outperforms the ANS method by a substantial margin, which ranges from a factor of 8 to 44. It is interesting to note that while the computing time of the ANPPA grows only modestly when the required accuracy tolerance is reduced by a factor of 10, the computing time for the ANS method grows by at least a factor of 2.

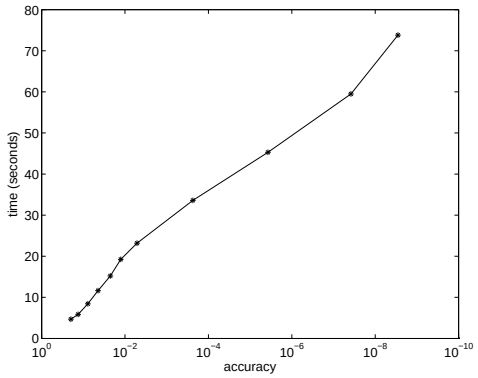

Figure 1: Accuracy versus time for the random instance `rand-500` reported in Table 3.

Table 4: Comparison of the ANPPA and the ANS method on (4) with synthetic data (II).

problem	$m \mid n$	tolerance		iteration		gap		time	
		PPA (Tol)	ANS (ϵ_o, ϵ_c)	PPA	ANS	PPA	ANS	PPA	ANS
rand-500	112172 | 500	3×10^{-6}	$(10^{-1}, 10^{-4})$	27	2358	5.53-5	9.79-2	61.4	518.3
		3×10^{-7}	$(10^{-2}, 10^{-4})$	27	5779	5.53-5	9.94-3	61.4	1233.2
		3×10^{-8}	$(10^{-3}, 10^{-4})$	30	12796	9.04-8	9.94-4	74.1	2712.3
rand-1000	441294 | 1000	3×10^{-6}	$(10^{-1}, 10^{-4})$	29	3776	3.97-3	9.91-2	468.8	4499.8
		3×10^{-7}	$(10^{-2}, 10^{-4})$	33	10055	3.86-6	9.94-3	583.5	11715.4
		3×10^{-8}	$(10^{-3}, 10^{-4})$	33	22519	3.86-6	9.94-4	583.5	26173.1
rand-1500	979620 | 1500	3×10^{-6}	$(10^{-1}, 10^{-4})$	29	3691	4.10-3	9.94-2	1384.5	13601.7
		3×10^{-7}	$(10^{-2}, 10^{-4})$	33	9027	1.05-4	9.94-3	1699.4	32440.2
		3×10^{-8}	$(10^{-3}, 10^{-4})$	33	18408	1.05-4	9.94-4	1699.4	65773.7

In the second synthetic experiment, we consider the problem (5). We set $\rho_{ij} = 1/n^{1.5}$ for all $(i, j) \notin \Omega$. We note that the parameters ρ_{ij} are chosen empirically so as to give a reasonably good value for $\|\Sigma - \widehat{\Sigma}\|_F$.

In Tables 5 and 6, we report the results in a similar format as those appeared in Table 3 and 4, respectively. Again, we may observe from the tables that the ANPPA outperformed the ANS method by a substantial margin.

Table 5: Performance of the ANPPA on (5) with synthetic data (II).

problem	$m \mid n$	*it/itsub/pcg*	pobj	dobj	$R_P/R_D/R_G/L_2^{\text{loss}}$	time
rand-500	112172 | 500	15| 50| 12.5	-3.11255742 2	-3.11256007 2	1.9-8| 1.7-7| 4.2-7| 1.7-2	105.7
rand-1000	441294 | 1000	17| 60| 18.3	-9.70441465 2	-9.70441034 2	2.1-8| 1.4-7| 2.2-7| 2.0-2	942.2
rand-1500	979620 | 1500	18| 55| 16.3	-1.90500086 3	-1.90499588 3	8.4-8| 4.7-7| 1.3-6| 1.8-2	2498.9
rand-2000	1719589 | 2000	19| 53| 16.8	-2.99725089 3	-2.99724734 3	1.9-8| 4.4-7| 5.9-7| 1.5-2	5429.7

Table 6: Comparison of the ANPPA and the ANS method on (5) with synthetic data (II).

problem	$m \mid n$	tolerance		iteration		gap		time	
		PPA (Tol)	ANS (ϵ_o, ϵ_c)	PPA	ANS	PPA	ANS	PPA	ANS
rand-500	112172 | 500	3×10^{-6}	$(10^{-1}, 10^{-4})$	50	2327	2.65-4	9.90-2	105.7	510.3
		3×10^{-7}	$(10^{-2}, 10^{-4})$	50	5818	2.65-4	9.94-3	105.7	1236.9
		3×10^{-8}	$(10^{-3}, 10^{-4})$	55	12962	6.65-7	9.94-4	121.4	2747.0
rand-1000	441294 | 1000	3×10^{-6}	$(10^{-1}, 10^{-4})$	60	3741	4.31-4	9.88-2	942.2	4460.8
		3×10^{-7}	$(10^{-2}, 10^{-4})$	60	9931	4.31-4	9.94-3	942.2	11562.8
		3×10^{-8}	$(10^{-3}, 10^{-4})$	65	22620	1.88-5	9.94-4	1090.4	26278.6
rand-1500	979620 | 1500	3×10^{-6}	$(10^{-1}, 10^{-4})$	55	3745	4.98-3	9.90-2	2498.9	13830.3
		3×10^{-7}	$(10^{-2}, 10^{-4})$	59	9572	2.25-4	9.94-3	2855.2	34208.0
		3×10^{-8}	$(10^{-3}, 10^{-4})$	59	20895	2.25-4	9.94-4	2855.2	73997.1

6.4 Real data experiments

In this part, we compare the ANPPA and the ANS method on two gene expression data sets. Since [2] had already considered these data sets, we can refer to [2] for the choice of the parameters ρ_{ij}.

6.4.1 Rosetta Inpharmatics Compendium

We applied our ANPPA and the ANS method to the Rosetta Inpharmatics Compendium of gene expression profiles described by Hughes et al. [10]. The data set contains 253 samples with $n = 6136$ variables. We aim to estimate the sparse covariance matrix of a Gaussian graphic model whose conditional independence is unknown. Naturally, we formulate it as the problem (5), with $\Omega = \emptyset$. As for the parameters, we set $\rho_{ij} = 0.0313$ as in [2].

As our ANPPA can only handle problems with matrix dimensions up to about 3000, we only test on a subset of the data. We create 3 subsets by taking 500, 1000, and 1500 variables with the highest variances, respectively. Note that as the variances vary widely, we normalized the sample covariance matrices to have unit variances on the diagonal.

In the experiments, we set $\texttt{Tol} = 10^{-6}$ for the ANPPA, and $(\epsilon_o, \epsilon_c) = (10^{-2}, 10^{-6})$ for the ANS method.

The performances of the ANPPA and ANS methods for the Rosetta Inpharmatics Compendium of gene expression profiles are presented in Table 7. From Table 7, we can see that although both methods can solve the problem, the ANPPA is about 3.7 times faster than the ANS method when $n = 1500$.

Table 7: Comparison of the ANPPA and ANS method on (5) with $\Omega = \emptyset$ for the Rosetta Inpharmatics Compendiuma data.

problem	$m \mid n$	tolerance		iteration		primal objective value		time	
		PPA (Tol)	ANS (ϵ_o)	PPA	ANS	PPA	ANS	PPA	ANS
Rosetta	| 500	10^{-6}	10^{-3}	40	636	-7.42642518 2	-7.42642052 2	57.0	127.6
Rosetta	| 1000	10^{-6}	10^{-3}	40	763	-1.66546366 3	-1.66546478 3	298.8	881.6
Rosetta	| 1500	10^{-6}	10^{-3}	42	972	-2.64937351 3	-2.64937721 3	914.4	3424.7

6.4.2 Iconix Microarray data

Next we analyze the performances of the ANPPA and ANS methods on a subset of a 10000 gene microarray data set obtained from 255 drug treated rat livers; see Natsoulis et al. [16] for details. In our first test problem, we take 200 variables with the highest variances from the large set to form the sample covariance matrix S. The other two test problems are created by considering 500 and 1000 variables with the highest variances in the large data set. As in the last data set, we normalized the sample covariance matrices to have unit variances on the diagonal.

For the same reason as aforementioned, we set $\Omega = \emptyset$ in the problem (5). We set $\rho_{ij} = 0.0853$ as in [2].

The performance of the ANPPA and ANS methods for the Iconix microarray data is presented in Table 8. From the table, we see that the ANPPA is about 5 times faster than the ANS method when $n = 1000$.

Table 8: Comparison of the ANPPA and ANS method on (5) with $\Omega = \emptyset$ for the Iconix microarray data.

problem	$m \mid n$	tolerance		iteration		primal objective value		time	
		PPA (Tol)	ANS (ϵ_o)	PPA	ANS	PPA	ANS	PPA	ANS
Iconix	| 200	10^{-6}	10^{-3}	43	1805	-6.13127781 0	-6.13036186 0	17.7	50.7
Iconix	| 500	10^{-6}	10^{-3}	54	3809	5.31683802 1	5.31688551 1	222.9	795.2
Iconix	| 1000	10^{-6}	10^{-3}	65	6646	1.78893452 2	1.78892330 2	1585.2	7847.3

7 Concluding remarks

We designed a primal PPA to solve log-det optimization problems. Rigorous convergence results for the PPA are obtained from the classical results for a generic proximal point algorithm. We also considered to accelerate the outer iteration of PPA by Newton's method and designed an alternating Newton primal PPA. Extensive numerical experiments conducted on log-det problems arising from sparse estimation of inverse covariance matrices in Gaussian graphical models using synthetic data and real data demonstrated that our ANPPA is very efficient.

In contrast to the case for the linear SDPs, the log-det term used in this paper plays a key role of a smoothing term such that the standard smooth Newton method can be used to solve the inner problem. The key discovery of this paper is the connection of the log-det smoothing term with the technique of using the squared smoothing function. It opens up a new door to deal with nonsmooth equations and understand the smoothing technique more deeply.

Acknowledgements

We thank Onureena Banerjee for providing us with part of the test data and helpful suggestions and Zhaosong Lu and Xiaoming Yuan for sharing with us their MATLAB codes and fruitful discussions. We also thank the two anonymous referees and the associate editor for their helpful comments and suggestions which improved the quality of this paper.

References

[1] F. Alizadeh, J. P. A. Haeberly, and O. L. Overton, *Complementarity and nondegeneracy in semidefinite programming*, Mathemtical Programming, 77 (1997), 111–128.

[2] O. Banerjee, L. El Ghaoui, A. d'Aspremont, *Model selection through sparse maximum likelihood estimation*, Journal of Machine Learning Research, 9 (2008), 485–516.

[3] R. Bhatia, *Matrix Analysis*, Springer-Verlag, New York, 1997.

[4] J. F. Bonnans and A. Shapiro, *Perturbation Analysis of Optimization Problems*, Springer, New York, 2000.

[5] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*, vol. 15 of Studies in Applied Mathematics, SIAM, Philadelphia, PA, 1994.

[6] Z. X. Chan and D. F. Sun, *Constraint nondegeneracy, strong regularity and nonsigularity in semidefinite programming*, SIAM Journal on optimization, 19 (2008), 370–396.

[7] J. Dahl, L. Vandenberghe, and V. Roychowdhury, *Covariance selection for nonchordal graphs via chordal embedding*, Optimization Methods and Software, 23 (2008), 501–520.

[8] A. d'Aspremont, O. Banerjee, and L. El Ghaoui, *First-order methods for sparse covariance selection*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), 56–66.

[9] Y. Gao and D. F. Sun, *Calibrating least squares covariance matrix problems with equality and inequality constraints*, SIAM Journal on Matrix Analysis and Applications, 31 (2009), 1432–1457.

[10] T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, M. J. Kidd, A. M. King, M. R. Meyer, D. Slade, P. Y. Lum, S. B. Stepaniants, D. D. Shoemaker, D. Gachotte,

K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend, *Functional discovery via a compendium of expression profiles*, Cell, 102(1) 2000, 109C-126.

[11] Z. Lu, *Smooth optimization approach for sparse covariance selection*, SIAM Journal on Optimization, 19(4) (2009), 1807–1827.

[12] Z. Lu, *Adaptive first-order methods for general sparse inverse covariance selection*, to appear in SIAM Journal on Matrix Analysis and Applications.

[13] F. Meng, D. F. Sun, and G. Zhao, *Semismoothness of solutions to generalized equations and the Moreau-Yosida regularization*, Mathemtical Programming, 104 (2005), 561–581.

[14] G. J. Minty, *On the monotonicity of the gradient of a convex function*, Pacific Journal of Mathematics, 14 (1964), 243–247.

[15] J. J. Moreau, *Proximité et dualité dans un espace Hilbertien*, Bulletin de la Société Mathématique de France, 93 (1965), 273–299.

[16] Georges Natsoulis, Cecelia I Pearson, Jeremy Gollub, Barrett P Eynon, Joe Ferng, Ramesh Nair, Radha Idury, May D Lee, Mark R Fielden, Richard J Brennan, Alan H Roter and Kurt Jarnagin, *The liver pharmacological and xenobiotic gene response repertoire*, Molecular Systems Biology, 175(4) (2008), 1–12.

[17] Yu. E. Nesterov, *Smooth minimization of nonsmooth functions*, Mathemtical Programming, 103 (2005), 127–152.

[18] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, 1970.

[19] R. T. Rockafellar, *Monotone operators and the proximal point algorithm*, SIAM Journal on Control and Optimization, 14 (1976), 877–898.

[20] R. T. Rockafellar, *Augmented Lagrangains and applications of the proximal point algorithm in convex programming*, Mathematics of Operation Research, 1 (1976), 97–116.

[21] R. T. Rockfellar and R. J. B. Wets, *Variational Analysis*, Springer, Berlin, 1998.

[22] K. Scheinberg and I. Rish, *SINCO - a greedy coordinate ascent method for sparse inverse covariance selection problem*, http://www.optimization-online.org/DB_FILE/2009/07/2359.pdf.

[23] D. F. Sun, J. Sun and L. W. Zhang, *The rate of convergence of the augmented Lagrangian method for nonlinear semidefinite programming*, Mathematical Programming, 114 (2008), 349–391.

[24] K .C. Toh, *Primal-dual path-following algorithms for determinant maximization problems with linear matrix inequalities*, Computational Optimization and Applications, 14 (1999), 309–330.

[25] R. H. Tütüncü, K. C. Toh, and M. J. Todd, *Solving semidefinite-quadratic-linear programs using SDPT3*, Mathemtical Programming 95 (2003), 189–217.

[26] N. -K. Tsing, M. K. H. Fan, and E. I. Verriest, *On analyticity of functions involving eigenvalues*, Linear Algebra and its Applications 207 (1994), 159–180.

[27] L. Vandenberghe, S. Boyd, and S. -P. Wu, *Determinant maximization with linear matrix inequality equalities*, SIAM Journal on Matrix Analysis and Applications, 19 (1998), 499–533.

[28] W. B. Wu, M. Pourahmadi, *Nonparameteric estimation of large covariance matrices of longitudinal data*, Biometrika, 90 (2003), 831–844.

[29] F. Wong, C. K. Carter, and R. Kohn, *Efficient estimation of covariance selection models*, Biometrika, 90 (2003), 809–830.

[30] K. Yosida, Functional Analysis, Springer Verlag, Berlin, 1964.

[31] X. M. Yuan, *Alternating direction methods for sparse covariance selection*, www.optimization-online.org/DB_FILE/2009/09/2390.pdf.

[32] X. Y. Zhao, D. F. Sun and K. C. Toh, *A Newton-CG augmented Lagrangian method for semidefinite programming*, SIAM Journal on Optimization, 20 (2010), 1737–1765.