Research Reports on Mathematical and Computing Sciences
Series B : Operations Research

Department of Mathematical and Computing Sciences
Tokyo Institute of Technology
2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan

# Interior-Point Methods for Lagrangian Duals of Semidefinite Programs

Mituhiro Fukuda (*mituhiro@is.titech.ac.jp*)

Masakazu Kojima (*kojima@is.titech.ac.jp*)
Department of Mathematical and Computing Sciences, Tokyo Institute of Technology
2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan

December 2000, B-365
Revised December 23

**Abstract** This paper proposes a new primal-dual predictor-corrector interior-point method for a class of semidefinite programs, which numerically traces the central trajectory in a space of Lagrange multipliers. The distinguishing features of the method are full use of the BFGS quasi-Newton method in the corrector procedure and an application of the conjugate gradient method with an effective preconditioning matrix induced from the BFGS quasi-Newton method in the predictor procedure. Some preliminary numerical results are presented.

**Key words.** Semidefinite Program, Linear Program over Symmetric Cones, Second-Order Cone Program, Primal-Dual Interior-Point Method, Predictor-Corrector Method, Lagrangian Dual, Central Trajectory, BFGS Quasi-Newton Method, Conjugate Gradient Method, Conjugate Residual Method.

# 1 Introduction.

Consider the equality standard form SDP and its dual

$$\left.\begin{array}{lll}\text{Primal SDP:} & \text{minimize} & \boldsymbol{A}_0 \bullet \boldsymbol{X} \\ & \text{subject to} & \boldsymbol{A}_p \bullet \boldsymbol{X} = b_p \ (p = 1, 2, \ldots, m), \ \boldsymbol{X} \in \mathrm{S}^n_+,\end{array}\right\} \tag{1}$$

$$\left.\begin{array}{lll}\text{Dual SDP:} & \text{maximize} & \sum_{p=1}^m b_p y_p \\ & \text{subject to} & \sum_{p=1}^m \boldsymbol{A}_p y_p + \boldsymbol{S} = \boldsymbol{A}_0, \ \boldsymbol{S} \in \mathrm{S}^n_+,\end{array}\right\} \tag{2}$$

where $\mathrm{S}^n_+$ denotes the cone of positive semidefinite matrices in the space $\mathrm{S}^n$ of $n \times n$ real symmetric matrices, $\boldsymbol{A}_p \in \mathrm{S}^n$ $(p = 0, 1, 2, \ldots, m)$ given matrices, $b_p \in \mathrm{R}$ $(p = 1, 2, \ldots, m)$ given real numbers, and $\boldsymbol{A} \bullet \boldsymbol{X}$ the inner product of $\boldsymbol{A} \in \mathrm{S}^n$ and $\boldsymbol{X} \in \mathrm{S}^n$ (*i.e.*, $\boldsymbol{A} \bullet \boldsymbol{X} = \text{Trace } \boldsymbol{A}^T \boldsymbol{X} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} X_{ij}$). We will also use the notation $\mathrm{R}_{++} \subset \mathrm{R}$ and $\mathrm{S}^n_{++} \subset \mathrm{S}^n_+$ for the set of positive numbers and the cone of positive definite matrices, respectively.

Primal-dual interior-point methods [1, 15, 18, 20, 28, etc.] for SDPs have been getting popular, and now several software packages [3, 8, 27, 30, etc.] based on them are available through Internet. However those software packages are not powerful enough to solve large scale general SDPs, *e.g.,* SDPs with m and/or n larger than several thousand. Serious difficulties arise when we solve the key linear equation $\boldsymbol{M}\boldsymbol{dy} = \boldsymbol{r}$ with a fully dense $m \times m$ matrix $\boldsymbol{M}$, which is often called the Schur complement equation, to obtain a search direction $(\boldsymbol{dX}, \boldsymbol{dy}, \boldsymbol{dS}) \in \mathrm{S}^n \times \mathrm{R}^m \times \mathrm{S}^n$. Here each component of the $m \times m$ coefficient matrix $\boldsymbol{M}$ depends not only on the data matrices $\boldsymbol{A}_p$ $(p = 1, 2, \ldots, m)$ but also on the iterate $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{S})$ at which the search direction $(\boldsymbol{dX}, \boldsymbol{dy}, \boldsymbol{dS})$ is computed. As $m$ gets larger, computing the coefficient matrix $\boldsymbol{M}$ becomes more expensive. When the data matrices are sparse and/or of rank not greater than two, some efficient methods were proposed for computing the coefficient matrix $\boldsymbol{M}$ [11, 13]. As $m$ gets larger, solving the equation $\boldsymbol{M}\boldsymbol{dy} = \boldsymbol{r}$ by direct methods such as the Cholesky factorization also becomes more expensive and more difficult. When $m$ is larger than ten thousand, it is even impossible to store the entire coefficient matrix $\boldsymbol{M}$ in standard workstations; hence we are forced to use iterative methods such as the CG (conjugate gradient) method and the CR (conjugate residual) method to solve the equation $\boldsymbol{M}\boldsymbol{dy} = \boldsymbol{r}$. However the condition number of the coefficient matrix $\boldsymbol{M}$ gets worse rapidly as the iterate $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{S})$ gets closer to an optimal solution in the primal-dual space. Under such circumstances, we need to improve the condition number of the coefficient matrix $\boldsymbol{M}$ by performing an appropriate preconditioning to $\boldsymbol{M}$ because otherwise even a lower accuracy solution of $\boldsymbol{M}\boldsymbol{dy} = \boldsymbol{r}$ would require more and more CG or CR iterations. It is a current important issue how we choose an effective preconditioning without storing the entire coefficient matrix $\boldsymbol{M}$. See [6, 19, 22, 29] for more details on applications of iterative methods, preconditioning and numerical experiments on some large scale SDPs.

Another inefficiency of using primal-dual interior-point methods is that the $n \times n$ primal matrix variable $\boldsymbol{X}$ is fully dense in general even when all the data matrices $\boldsymbol{A}_p \in \mathrm{S}^n$ $(p = 0, 1, 2, \ldots, m)$ are sparse. This is a disadvantage of primal-dual interior-point methods compared to the dual scaling method [2] which generates iterates only in the dual space; note that the dual matrix variable $\boldsymbol{S}$ computed by $\boldsymbol{S} = \boldsymbol{A}_0 - \sum_{p=1}^m \boldsymbol{A}_p y_p$ inherits the sparsity of the data matrices $\boldsymbol{A}_p \in \mathrm{S}^n$ $(p = 0, 1, 2, \ldots, m)$. To overcome this disadvantage, Fukuda-Kojima-Murota-Nakata [12] recently proposed methods based on the positive definite matrix completion for exploiting the aggregate sparsity pattern over the data matrices.

Besides interior-point methods, some other computational methods have been also proposed and intensively studied for solving large scale SDPs; the spectral bundle method [14] and nonlinear programming reformulation of SDPs [4, 5, 34].

As we have mentioned above, lots of efforts have been in progress to solve large scale SDPs from various different angles. Numerical results on large scale SDPs have been reported. They include (i) SDP relaxations of the max cut problem and the graph bisection problem solved by the spectral bundle method [13, 14], the dual-scaling method [2] and nonlinear programming [4, 5], (ii) an SDP relaxation of the max clique problem by the primal-dual interior-point method with the use of the CG method [22] and the CR method [29]. However successful numerical results on large scale SDPs have been restricted so far to a few types of such SDPs (arising from SDP relaxation of combinatorial optimization problems on graph) that do not require highly accurate solutions.

"Solving general large scale SDPs in high accuracy" still remains a significant, unsettled and challenging subject in mathematical programming. Aiming to resolve many of the difficulties mentioned so far, this paper proposes a new computational method for solving a class of SDPs. We are concerned with the following SDP and its dual.

$$\left.\begin{array}{lll}\text{Primal SDP:} & \text{minimize} & \boldsymbol{A}_0 \bullet \boldsymbol{X} \\ & \text{subject to} & \boldsymbol{A}_p \bullet \boldsymbol{X} = b_p \ (p = 1, 2, \ldots, m), \\ & & \boldsymbol{I} \bullet \boldsymbol{X} = b_{m+1}, \ \ \boldsymbol{X} \in \mathrm{S}^n_+. \end{array}\right\} \tag{3}$$

$$\left.\begin{array}{lll}\text{Dual SDP:} & \text{maximize} & \sum_{p=1}^{m+1} b_p y_p \\ & \text{subject to} & \sum_{p=1}^{m} \boldsymbol{A}_p y_p + \boldsymbol{I} y_{m+1} + \boldsymbol{S} = \boldsymbol{A}_0, \ \ \boldsymbol{S} \in \mathrm{S}^n_+. \end{array}\right\} \tag{4}$$

Here $\boldsymbol{I}$ denotes the $n \times n$ identity matrix and $b_{m+1}$ a positive number. Throughout the paper we assume:

**Condition 1.1.**

*(i) There is an interior feasible solution $\boldsymbol{X}$ of (3), i.e., an $\boldsymbol{X} \in \mathrm{S}^n_{++}$ which satisfies the constraints of (3).*

*(ii) The data matrices $\boldsymbol{A}_p$ $(p = 1, 2, \ldots, m)$ and $\boldsymbol{I}$ which appear in the equality constraints of (3) are linearly independent.*

Note that the primal SDP (3) involves a "simplex constraint" $\boldsymbol{X} \in \Omega_+ \equiv \{\boldsymbol{X} \in \mathrm{S}^n_+ : \boldsymbol{I} \bullet \boldsymbol{X} = b_{m+1}\}$.We will denote the interior $\{\boldsymbol{X} \in \mathrm{S}^n_{++} : \boldsymbol{I} \bullet \boldsymbol{X} = b_{m+1}\}$ of $\Omega_+$ by $\Omega_{++}$. Although this constraint is restrictive, the primal SDP (3) covers various and important SDPs such as SDP relaxations of combinatorial optimization problems. We also note that if the feasible region of a given SDP (1) without the simplex constraint is bounded and a bound is known in advance, then we can transform it into the primal SDP (3) above. Helmberg and Rendl [14] dealt with this type of SDP for which they presented their spectral bundle method.

The basic idea behind our method is to trace the central trajectory in the space of dual variables $y_1, y_2, \ldots, y_m$. Based on the Lagrangian duality theory, we will construct a family of strictly convex smooth functions $g(\cdot; \mu) : \mathrm{R}^m \rightarrow \mathrm{R}$ $(\mu \in \mathrm{R}_{++})$ whose minimizers over the entire space $\mathrm{R}^m$ form the central trajectory in the space $\mathrm{R}^m$ of Lagrange multipliers

$y_1, y_2, \ldots, y_m$; if we denote the central trajectory of the primal-dual pair of SDPs (3) and (4) by $\mathcal{C} = \{(\boldsymbol{X}(\mu), \boldsymbol{y}(\mu), y_{m+1}(\mu), \boldsymbol{S}(\mu), \mu) : \mu \in \mathrm{R}_{++}\}$, then for every $\mu > 0$, $\boldsymbol{y} \in \mathrm{R}^m$ is a minimizer of $g(\cdot; \mu)$ over $\mathrm{R}^m$ if and only if $\boldsymbol{y} = \boldsymbol{y}(\mu)$. Here $\mu \in \mathrm{R}_{++}$ denotes the barrier parameter.

Our method is a predictor-corrector method that numerically traces the trajectory $\{(\boldsymbol{y}(\mu), \mu) : \mu \in \mathrm{R}_{++}\}$. Let $\bar{\mu} \in \mathrm{R}_{++}$. The role of the corrector procedure is to approximate the minimizer $\boldsymbol{y}(\bar{\mu})$ of the function $g(\cdot; \bar{\mu})$ over $\mathrm{R}^m$ starting from a $\hat{\boldsymbol{y}} \in \mathrm{R}^m$. Here we assume that the point $(\hat{\boldsymbol{y}}, \bar{\mu})$ has been generated by the predictor procedure of the previous iteration or given initially in the first iteration. Let $\boldsymbol{y}^0 = \hat{\boldsymbol{y}}, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^k, \ldots$ be a sequence generated by the corrector procedure in the space $\mathrm{R}^m$ of the Lagrange multiplier vector $\boldsymbol{y}$. In generating the sequence,

**(a)** we can fully utilize various unconstrained optimization methods such as quasi-Newton methods with the use of first derivatives.

In connection with our predictor procedure later, it is convenient for us to use the BFGS quasi-Newton method which updates an approximation $\boldsymbol{H}^k$ of the inverse of the Hessian matrix $\boldsymbol{\nabla}^2 g(\boldsymbol{y}^k; \bar{\mu})$ (or the Cholesky factorization $\boldsymbol{B}^k = \boldsymbol{L}^k(\boldsymbol{L}^k)^T$ of an approximation $\boldsymbol{B}^k$ of the Hessian matrix $\boldsymbol{\nabla}^2 g(\boldsymbol{y}^k; \bar{\mu})$ itself of the function $g(\cdot; \bar{\mu})$ at $\boldsymbol{y} = \boldsymbol{y}^k$. We can also utilize the limited memory BFGS quasi-Newton method [24] when the dimension $m$ of the $\boldsymbol{y}$-space is large.

In spite of the nice feature (a), it is not so easy to numerically approximate the minimizer $\boldsymbol{y}(\mu)$ for small values of the barrier parameter $\mu \in \mathrm{R}_{++}$. This is because the condition number of the Hessian matrix $\boldsymbol{\nabla}^2 g(\cdot; \mu)$ gets worse rapidly as the barrier parameter $\mu$ gets smaller. We need to start with a larger barrier parameter $\mu$, and then we need to trace the curve $\{(\boldsymbol{y}(\mu), \mu) : \mu \in \mathrm{R}_{++}\}$ in a careful way. In practical implementation, it is also important to employ globally convergent and stable unconstrained optimization methods.

When we evaluate the function value $g(\boldsymbol{y}^k; \bar{\mu})$ and the gradient vector $\boldsymbol{\nabla} g(\boldsymbol{y}^k; \bar{\mu})$ at each iteration of the corrector procedure, we simultaneously generate $\boldsymbol{X}^k \in \mathrm{S}^n_{++}$ in the primal space, and $(y^k_{m+1}, \boldsymbol{S}^k) \in \mathrm{R} \times \mathrm{S}^n_{++}$ in the dual space. Each $(\boldsymbol{y}^k, y^k_{m+1}, \boldsymbol{S}^k)$ is an interior feasible solution of the dual SDP (4), while each $\boldsymbol{X}^k \in \mathrm{S}^n_{++}$ in the primal space is infeasible in general. In addition, each primal-dual pair $(\boldsymbol{X}^k, \boldsymbol{S}^k)$ of matrix variables satisfies $\boldsymbol{X}^k \boldsymbol{S}^k = \bar{\mu}\boldsymbol{I}$ $(k = 0, 1, 2, \ldots)$. Thus,

**(b)** the variables $(\boldsymbol{X}, \boldsymbol{y}, y_{m+1}, \boldsymbol{S})$ of the primal-dual pair of SDPs (3) and (4) are evaluated only where $(\boldsymbol{y}, y_{m+1}, \boldsymbol{S})$ is an interior feasible solution of (4) and the relations $\boldsymbol{X}\boldsymbol{S} = \bar{\mu}\boldsymbol{I}$ holds.

This implies that the dual matrix variable $\boldsymbol{S}$ inherits the sparsity of the data matrices $\boldsymbol{A}_p$ $(p = 0, 1, 2, \ldots, m)$ as in the existing primal-dual interior-point methods and also the inverse $\boldsymbol{X}^{-1} = \boldsymbol{S}/\mu$ of the primal matrix variable $\boldsymbol{X}$ shares exactly the same sparsity with $\boldsymbol{S}$. Furthermore, throughout the iterations, we can avoid explicitly computing and maintaining the primal matrix variable $\boldsymbol{X}$ if we compute and maintain the (sparse) Cholesky factorization of $\boldsymbol{S}$.

We should mention, however, that

**(c)** the evaluation of the function $g(\cdot;\mu)$ (and also the gradient vector $\nabla g(\cdot;\mu)$) at $\bar{\boldsymbol{y}} \in \mathrm{R}^m$ usually requires more than one Cholesky factorization in the dual matrix variable subspace $\{\boldsymbol{S} = \boldsymbol{A}_0 - \sum_{p=1}^{m} \boldsymbol{A}_p \bar{y}_p - \boldsymbol{I} y_{m+1} : y_{m+1} \in \mathrm{R}\}$.

The cost of the Cholesky factorization of $\boldsymbol{S}$ deeply depends on the aggregated sparsity pattern of the data matrices $\boldsymbol{A}_0, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_p$. See [12]. When the dual matrix variable $\boldsymbol{S}$ is large scale and the aggregated sparsity pattern is dense, the cost of the Cholesky factorization of $\boldsymbol{S}$ is very expensive. So, in the development of our method, we implicitly assume that either the size of $\boldsymbol{S}$ (= the size of the primal matrix variable $\boldsymbol{X}$) is not large compared to the number $m+1$ of the data matrices (= the number of equality constraints in the primal SDP (3)) or the data matrices $\boldsymbol{A}_0, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_p$ enjoy a nice aggregated sparsity pattern that allows a cheap Cholesky factorization of $\boldsymbol{S}$.

Now, suppose that the $r$th iterate $\boldsymbol{y}^r$ of the corrector procedure attains an approximation of the minimizer $\boldsymbol{y}(\bar{\mu})$ of $g(\cdot;\bar{\mu})$ over $\mathrm{R}^m$. Then the point $(\boldsymbol{y}^r, \bar{\mu})$ lies approximately on the trajectory $\{(\boldsymbol{y}(\mu), \mu) : \mu \in \mathrm{R}_{++}\}$ or $\boldsymbol{y}^r \approx \boldsymbol{y}(\bar{\mu})$. We then perform the predictor procedure based on a second order model for approximating a point $(\boldsymbol{y}(\mu^+), \mu^+)$, with some positive $\mu^+$ less than the current $\bar{\mu}$, on the trajectory $\{(\boldsymbol{y}(\mu), \mu) : \mu \in \mathrm{R}_{++}\}$:

$$\boldsymbol{y}(\mu) \approx \boldsymbol{y}(\bar{\mu}) + \dot{\boldsymbol{y}}(\bar{\mu})(\mu - \bar{\mu}) + \frac{1}{2}\ddot{\boldsymbol{y}}(\bar{\mu})(\mu - \bar{\mu})^2.$$

Here $\dot{\boldsymbol{y}}(\bar{\mu})$ and $\ddot{\boldsymbol{y}}(\bar{\mu})$ denote the first and second derivatives of $\boldsymbol{y}(\mu)$ evaluated at $\mu = \bar{\mu}$, respectively. The derivatives $\dot{\boldsymbol{y}}(\bar{\mu})$ and $\ddot{\boldsymbol{y}}(\bar{\mu})$ are given as solutions of systems of linear equations. Those systems involve the Hessian matrix $\nabla^2 g(\boldsymbol{y}(\bar{\mu});\bar{\mu})$ of the function $g(\cdot;\bar{\mu})$ at $\boldsymbol{y} = \boldsymbol{y}(\bar{\mu})$ as a common coefficient matrix. Therefore, to solve the systems approximately, we can utilize the approximation $\boldsymbol{H}^r$, which has been computed in the previous corrector procedure, of the inverse of the Hessian matrix $\nabla^2 g(\boldsymbol{y}(\bar{\mu});\bar{\mu})$. Specifically, if we replace the inverse $\nabla^2 g(\boldsymbol{y}(\bar{\mu});\bar{\mu})^{-1}$ of the coefficient Hessian matrix by $\boldsymbol{H}^r$ in solving those systems, we obtain approximations $\boldsymbol{f} \in \mathrm{R}^m$ and $\boldsymbol{s} \in \mathrm{R}^m$ of $\dot{\boldsymbol{y}}(\bar{\mu})$ and $\ddot{\boldsymbol{y}}(\bar{\mu})$, respectively. Then we take a positive $\mu^+ < \bar{\mu}$ and generate a point

$$\boldsymbol{y}^+ = \boldsymbol{y}^r + \boldsymbol{f}(\bar{\mu})(\mu^+ - \bar{\mu}) + \frac{1}{2}\boldsymbol{s}(\mu^+ - \bar{\mu})^2.$$

The point $(\boldsymbol{y}^+, \mu^+)$ will serve as the initial point of the succeeding corrector procedure for approximating the minimizer $\boldsymbol{y}(\mu^+)$ of the function $g(\cdot;\mu^+)$ over $\mathrm{R}^m$ or approximating the point $(\boldsymbol{y}(\mu^+), \mu^+)$ on the trajectory $\{(\boldsymbol{y}(\mu), \mu) : \mu \in \mathrm{R}_{++}\}$.

The computational efficiency of our method deeply relies on how accurately we can approximate the derivatives $\dot{\boldsymbol{y}}(\bar{\mu})$ and $\ddot{\boldsymbol{y}}(\bar{\mu})$; if the accuracy of their approximation is worse, not only we are forced to take a shorter predictor step length $\bar{\mu} - \mu^+$ but also the second order model introduced above has less meaning. In order to get more accurate approximation of the derivatives $\dot{\boldsymbol{y}}(\bar{\mu})$ and $\ddot{\boldsymbol{y}}(\bar{\mu})$, we apply iterative methods such as the CG (conjugate gradient) method and the CR (conjugate residual) method to the systems of equations in $\dot{\boldsymbol{y}}(\bar{\mu})$ and in $\ddot{\boldsymbol{y}}(\bar{\mu})$. It should be emphasized here that

**(d)** the matrix $\boldsymbol{H}^r$ serves as a powerful preconditioning matrix whenever $\boldsymbol{H}^r$ approximate the inverse of the coefficient Hessian matrix $\nabla^2 g(\boldsymbol{y}(\bar{\mu});\bar{\mu})$ with a reasonable accuracy.

This preconditioning technique is essential to make our predictor procedure more effective. The paper [21] proposes a preconditioning of the CG method with the use of the limited memory BFGS update [24].

In addition to the distinguished features (a), (b), (c) and (d) of our method,

**(e)** our method is a variation of primal-dual interior-point methods [1, 15, 18, 20, 28, etc.] for SDPs.

Among others, we mention that our method can be extended to more general class of linear optimization problems over symmetric cones including second-order programming problems in a straightforward way. See remark (C) in Section 5.

We must mention one major weakness of our method which we need to overcome in practical implementation. That is,

**(f)** the condition number of the Hessian matrix $\nabla^2 g(\cdot;\mu)$ of the objective function $g(\cdot;\mu)$ to be minimized over $\mathrm{R}^m$ gets worse rapidly when the barrier parameter $\mu$ gets smaller.

This means that the approximation of the minimizer $\boldsymbol{y}(\mu)$ of $g(\cdot;\mu)$ over $\mathrm{R}^m$ in a higher accuracy is not an easy task at all.

We present some more technical details in Section 2, and our prototype method and its variation in Section 3. Section 4 is devoted to preliminary numerical results.

**Remark 1.2.** The basic idea of using Lagrangian duals in interior-point methods was originally proposed for LPs in the working paper [16] in 1993, and discussed in the succeeding paper [17]. However, some theoretical aspects of the method were mainly investigated, and none of the papers was published because the method would be unlikely to compete with the primal-dual interior-point method for LPs, which had already become a powerful computational method for solving large scale LPs at that time, and also because some proofs of the main theorem of [17] were incomplete. This was pointed out by Gonguin Zhao. He later proposed an implementable version [31] of interior-point methods based on Lagrangian duals of linear programs, and proved its polynomial-time computational complexity. See also [33, 32].

# 2 Basic analysis.

## 2.1 Lagrangian duals.

In this section, we construct a family of strictly convex smooth functions $g(\cdot;\mu) : \mathrm{R}^m \to \mathrm{R}$ ($\mu \in \mathrm{R}_{++}$) whose minimizers form the central trajectory in the space $\mathrm{R}^m$ of Lagrange multipliers $y_1, y_2, \ldots, y_m$, and present formulae for its gradient vector $\nabla g(\cdot;\mu) : \mathrm{R}^m \to \mathrm{R}^m$

and Hessian matrix $\nabla^2 g(\cdot;\mu) : \mathrm{R}^m \to \mathrm{S}^m$. We first introduce introduce a Lagrangian relaxation of the SDP (3). For every $\boldsymbol{y} \in \mathrm{R}^m$, consider

$$\left.\begin{array}{ll} \text{minimize} & \left(\boldsymbol{A}_0 - \sum_{p=1}^{m} \boldsymbol{A}_p y_p\right) \bullet \boldsymbol{X} + \sum_{p=1}^{m} b_p y_p \\ \text{subject to} & \boldsymbol{X} \in \Omega_+ \equiv \{\boldsymbol{X} \in \mathrm{S}^n_+ : \boldsymbol{I} \bullet \boldsymbol{X} = b_{m+1}\}. \end{array}\right\} \tag{5}$$

Adding the logarithmic barrier term $-\mu \log\det \boldsymbol{X}$ with $\mu \in \mathrm{R}_{++}$ to the objective function, we obtain

$$\left.\begin{array}{ll} \text{minimize} & f(\boldsymbol{y}, \boldsymbol{X}; \mu) \\ \text{subject to} & \boldsymbol{X} \in \Omega_{++} \equiv \{\boldsymbol{X} \in \mathrm{S}^n_{++} : \boldsymbol{I} \bullet \boldsymbol{X} = b_{m+1}\}, \end{array}\right\} \tag{6}$$

where

$$\begin{array}{rcl} f(\boldsymbol{y}, \boldsymbol{X}; \mu) & = & \left(\boldsymbol{A}_0 - \sum_{p=1}^{m} \boldsymbol{A}_p y_p\right) \bullet \boldsymbol{X} + \sum_{p=1}^{m} b_p y_p - \mu \log\det \boldsymbol{X} \\ & & \text{for every } (\boldsymbol{y}, \boldsymbol{X}, \mu) \in \mathrm{R}^m \times \mathrm{S}^n_{++} \times \mathrm{R}_{++}, \end{array} \tag{7}$$

and $\mu \in \mathrm{R}_{++}$ denotes the barrier parameter. For every $(\boldsymbol{y}, \mu) \in \mathrm{R}_{++} \times \mathrm{R}^m$, the problem (6) has a unique optimal solution, which we will denote by $\boldsymbol{X}(\boldsymbol{y};\mu)$. Furthermore the optimal solution $\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{y};\mu)$ is characterized as the optimality condition:

$$\boldsymbol{I} \bullet \boldsymbol{X} = b_{m+1},\ \boldsymbol{I} y_{m+1} + \boldsymbol{S} = \boldsymbol{A}_0 - \sum_{p=1}^{m} \boldsymbol{A}_p y_p,\ \boldsymbol{X}\boldsymbol{S} = \mu \boldsymbol{I}, \boldsymbol{X} \in \mathrm{S}^n_{++},\ \boldsymbol{S} \in \mathrm{S}^n_{++}; \tag{8}$$

$\boldsymbol{X} \in \mathrm{S}^n$ is an optimal solution of the problem (6) if and only if it satisfies (8) for some $(y_{m+1}, \boldsymbol{S}) \in \mathrm{R} \times \mathrm{S}^n$. We will show later how to compute $(\boldsymbol{X}, y_{m+1}, \boldsymbol{S}) \in \mathrm{S}^n \times \mathrm{R} \times \mathrm{S}^n$ satisfying (8).

Now we define the function $g(\cdot;\mu) : \mathrm{R}^m \to \mathrm{R}$ as the optimal value of the problem (6); for every $(\boldsymbol{y};\mu) \in \mathrm{R}^m \times \mathrm{R}_{++}$,

$$g(\boldsymbol{y};\mu) \ \equiv\ -\min\{f(\boldsymbol{y}, \boldsymbol{X}; \mu) : \boldsymbol{X} \in \Omega_{++}\} = -f(\boldsymbol{y}, \boldsymbol{X}(\boldsymbol{y};\mu);\mu).$$

Suppose that $\boldsymbol{X} \in \mathrm{S}^n_{++}$ and $\boldsymbol{S} \in \mathrm{S}^n_{++}$. We will often use the following notation.

$$\left.\begin{array}{lcl} \boldsymbol{A}_{m+1} & \equiv & \boldsymbol{I}, \\ B_{qr} & \equiv & \boldsymbol{A}_q \bullet \boldsymbol{X} \boldsymbol{A}_r \boldsymbol{S}^{-1} \\ & & (q = 1, 2, \ldots, m+1,\ r = 1, 2, \ldots, m+1), \\ \boldsymbol{c} & \equiv & (B_{1,m+1}, B_{2,m+1}, \ldots, B_{m,m+1})^T, \\ \boldsymbol{C} & \equiv & \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1m} \\ B_{21} & B_{22} & \cdots & B_{2m} \\ \cdots & \cdots & \ddots & \cdots \\ B_{m1} & B_{m2} & \cdots & B_{mm} \end{pmatrix} \in \mathrm{S}^m, \\ \boldsymbol{B} & \equiv & \begin{pmatrix} \boldsymbol{C} & \boldsymbol{c} \\ & \\ \boldsymbol{c}^T & B_{m+1,m+1} \end{pmatrix} \in \mathrm{S}^{1+m}. \end{array}\right\} \tag{9}$$

The matrix $\boldsymbol{B}$ above corresponds to the coefficient matrix of the so-called Schur complement equation for the HRVW/KSH/M search direction [15, 18, 20]. In particular, the matrix $\boldsymbol{B}$ is known to be symmetric and positive definite. It should be also noted that when the primal-dual pair of matrix variables $\boldsymbol{X} \in \mathbb{S}^n_{++}$ and $\boldsymbol{S} \in \mathbb{S}^n_{++}$ satisfies $\boldsymbol{X}\boldsymbol{S} = \mu \boldsymbol{I}$ as in the succeeding discussions, the matrix $\boldsymbol{B}$ is also identical to the coefficient matrix of the Schur complement equation for other well-known search directions including the AHO direction [1] and the NT-direction [28].

**Lemma 2.1.**

*(i)* $\nabla g(\boldsymbol{y};\mu) = \left(\boldsymbol{A}_1 \bullet \boldsymbol{X}(\boldsymbol{y};\mu) - b_1, \ldots, \boldsymbol{A}_m \bullet \boldsymbol{X}(\boldsymbol{y};\mu) - b_m\right)^T$.

*(ii)* $\nabla^2 g(\boldsymbol{y};\mu) = \left(\boldsymbol{C} - \boldsymbol{c}\boldsymbol{c}^T / B_{m+1,m+1}\right)$. *Here* $\boldsymbol{C} \in \mathbb{S}^m$, $\boldsymbol{c} \in \mathbb{R}^m$ *and* $B_{m+1,m+1} \in \mathbb{R}$ *are given by (9) with* $\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{y};\mu)$ *and* $\boldsymbol{S} = \boldsymbol{S}(\boldsymbol{y};\mu)$.

*(iii) The Hessian matrix* $\nabla^2 g(\boldsymbol{y};\mu)$ *is positive definite.*

*Proof:* First we observe that $(\boldsymbol{X}(\boldsymbol{y};\mu), y_{m+1}(\boldsymbol{y};\mu), \boldsymbol{S}(\boldsymbol{y};\mu)) \in \mathbb{S}^n_{++} \times \mathbb{R} \times \mathbb{S}^n_{++}$ satisfies

$$\left.\begin{aligned} &\boldsymbol{I} \bullet \boldsymbol{X}(\boldsymbol{y};\mu) = b_{m+1}, \\ &\boldsymbol{I} y_{m+1}(\boldsymbol{y};\mu) + \boldsymbol{S}(\boldsymbol{y};\mu) = \boldsymbol{A}_0 - \sum_{p=1}^m \boldsymbol{A}_p y_p, \\ &\boldsymbol{X}(\boldsymbol{y};\mu)\boldsymbol{S}(\boldsymbol{y};\mu) = \mu \boldsymbol{I}. \end{aligned}\right\} \tag{10}$$

Differentiating these identities by $y_q$, we obtain

$$\left.\begin{aligned} &\boldsymbol{I} \bullet \frac{\partial \boldsymbol{X}(\boldsymbol{y};\mu)}{\partial y_q} = 0, \\ &\boldsymbol{I}\frac{\partial y_{m+1}(\boldsymbol{y};\mu)}{\partial y_q} + \frac{\partial \boldsymbol{S}(\boldsymbol{y};\mu)}{\partial y_q} = -\boldsymbol{A}_q, \\ &\boldsymbol{X}(\boldsymbol{y};\mu)\frac{\partial \boldsymbol{S}(\boldsymbol{y};\mu)}{\partial y_q} + \frac{\partial \boldsymbol{X}(\boldsymbol{y};\mu)}{\partial y_q}\boldsymbol{S}(\boldsymbol{y};\mu) = \boldsymbol{O}. \end{aligned}\right\} \tag{11}$$

By (10), (11) and the chain rule, we obtain

$$\begin{aligned} &-\frac{\partial g(\boldsymbol{y};\mu)}{\partial y_q} \\ &= \left.\frac{\partial f(\boldsymbol{y},\boldsymbol{X};\mu)}{\partial y_q}\right|_{\boldsymbol{X}=\boldsymbol{X}(\boldsymbol{y};\mu)} + \sum_{i=1}^n \sum_{j=1}^n \left.\frac{\partial f(\boldsymbol{y},\boldsymbol{X};\mu)}{\partial X_{ij}}\right|_{\boldsymbol{X}=\boldsymbol{X}(\boldsymbol{y};\mu)} \times \frac{\partial X_{ij}(\boldsymbol{y};\mu)}{\partial y_q} \\ &= b_q - \boldsymbol{A}_q \bullet \boldsymbol{X}(\boldsymbol{y};\mu) + \left(\boldsymbol{A}_0 - \sum_{p=1}^m \boldsymbol{A}_p y_p - \mu \boldsymbol{X}(\boldsymbol{y};\mu)^{-1}\right) \bullet \frac{\partial \boldsymbol{X}(\boldsymbol{y};\mu)}{\partial y_q} \\ &= b_q - \boldsymbol{A}_q \bullet \boldsymbol{X}(\boldsymbol{y};\mu) + \left(\boldsymbol{A}_0 - \sum_{p=1}^m \boldsymbol{A}_p y_p - \boldsymbol{S}(\boldsymbol{y};\mu)\right) \bullet \frac{\partial \boldsymbol{X}(\boldsymbol{y};\mu)}{\partial y_q} \\ &= b_q - \boldsymbol{A}_q \bullet \boldsymbol{X}(\boldsymbol{y};\mu) + (\boldsymbol{I} y_{m+1}) \bullet \frac{\partial \boldsymbol{X}(\boldsymbol{y};\mu)}{\partial y_q} \\ &= b_q - \boldsymbol{A}_q \bullet \boldsymbol{X}(\boldsymbol{y};\mu). \end{aligned}$$

Thus we have shown (i). By (11), we also see that

$$\begin{aligned}
\frac{\partial \boldsymbol{X}(\boldsymbol{y};\mu)}{\partial y_q} &= -\boldsymbol{X}(\boldsymbol{y};\mu)\frac{\partial \boldsymbol{S}(\boldsymbol{y};\mu)}{\partial y_q}\boldsymbol{S}(\boldsymbol{y};\mu)^{-1} \\
&= \boldsymbol{X}(\boldsymbol{y};\mu)\left(\boldsymbol{I}\frac{\partial y_{m+1}(\boldsymbol{y};\mu)}{\partial y_q} + \boldsymbol{A}_q\right)\boldsymbol{S}(\boldsymbol{y};\mu)^{-1} \\
&= \frac{\partial y_{m+1}(\boldsymbol{y};\mu)}{\partial y_q}\boldsymbol{X}(\boldsymbol{y};\mu)\boldsymbol{S}(\boldsymbol{y};\mu)^{-1} + \boldsymbol{X}(\boldsymbol{y};\mu)\boldsymbol{A}_q\boldsymbol{S}(\boldsymbol{y};\mu)^{-1} \\
0 &= \boldsymbol{I} \bullet \frac{\partial \boldsymbol{X}(\boldsymbol{y};\mu)}{\partial y_q} \\
&= \frac{\partial y_{m+1}(\boldsymbol{y};\mu)}{\partial y_q}\boldsymbol{X}(\boldsymbol{y};\mu) \bullet \boldsymbol{S}(\boldsymbol{y};\mu)^{-1} + \boldsymbol{X}(\boldsymbol{y};\mu) \bullet \boldsymbol{A}_q\boldsymbol{S}(\boldsymbol{y};\mu)^{-1}.
\end{aligned}$$

It follows that

$$\begin{aligned}
\frac{\partial y_{m+1}(\boldsymbol{y};\mu)}{\partial y_q} &= -\frac{\boldsymbol{X}(\boldsymbol{y};\mu) \bullet \boldsymbol{A}_q\boldsymbol{S}(\boldsymbol{y};\mu)^{-1}}{\boldsymbol{X}(\boldsymbol{y};\mu) \bullet \boldsymbol{S}(\boldsymbol{y};\mu)^{-1}}, \\
\frac{\partial \boldsymbol{X}(\boldsymbol{y};\mu)}{\partial y_q} &= -\frac{\boldsymbol{X}(\boldsymbol{y};\mu) \bullet \boldsymbol{A}_q\boldsymbol{S}(\boldsymbol{y};\mu)^{-1}}{\boldsymbol{X}(\boldsymbol{y};\mu) \bullet \boldsymbol{S}(\boldsymbol{y};\mu)^{-1}}\boldsymbol{X}(\boldsymbol{y};\mu)\boldsymbol{S}(\boldsymbol{y};\mu)^{-1} + \boldsymbol{X}(\boldsymbol{y};\mu)\boldsymbol{A}_q\boldsymbol{S}(\boldsymbol{y};\mu)^{-1} \\
&= -\frac{\boldsymbol{A}_q \bullet \boldsymbol{X}(\boldsymbol{y};\mu)^2}{\mu\boldsymbol{X}(\boldsymbol{y};\mu) \bullet \boldsymbol{X}(\boldsymbol{y};\mu)}\boldsymbol{X}(\boldsymbol{y};\mu)^2 + \frac{\boldsymbol{X}(\boldsymbol{y};\mu)\boldsymbol{A}_q\boldsymbol{X}(\boldsymbol{y};\mu)}{\mu} \\
&\quad (\text{since } \boldsymbol{X}(\boldsymbol{y};\mu)\boldsymbol{S}(\boldsymbol{y};\mu) = \mu\boldsymbol{I}).
\end{aligned}$$

Hence we obtain

$$\begin{aligned}
&\frac{\partial^2 g(\boldsymbol{y};\mu)}{\partial y_r \partial y_q} \\
&= \boldsymbol{A}_q \bullet \frac{\partial \boldsymbol{X}(\boldsymbol{y};\mu)}{\partial y_r} \\
&= \boldsymbol{A}_q \bullet \left(\frac{\boldsymbol{X}(\boldsymbol{y};\mu)\boldsymbol{A}_r\boldsymbol{X}(\boldsymbol{y};\mu)}{\mu} - \frac{\boldsymbol{A}_q \bullet \boldsymbol{X}(\boldsymbol{y};\mu)^2}{\mu\boldsymbol{X}(\boldsymbol{y};\mu) \bullet \boldsymbol{X}(\boldsymbol{y};\mu)}\boldsymbol{X}(\boldsymbol{y};\mu)^2\right) \\
&= \frac{\boldsymbol{A}_q \bullet (\boldsymbol{X}(\boldsymbol{y};\mu)\boldsymbol{A}_r\boldsymbol{X}(\boldsymbol{y};\mu))}{\mu} - \frac{(\boldsymbol{A}_r \bullet \boldsymbol{X}(\boldsymbol{y};\mu)^2)(\boldsymbol{A}_q \bullet \boldsymbol{X}(\boldsymbol{y};\mu)^2)}{\mu\boldsymbol{X}(\boldsymbol{y};\mu) \bullet \boldsymbol{X}(\boldsymbol{y};\mu)}
\end{aligned}$$

Thus (ii) follows. Note that the matrix $\boldsymbol{C} - \boldsymbol{c}^T\boldsymbol{c}/B_{m+1,m+1}$ in the bracket in (ii) is a Schur complement of the matrix $\boldsymbol{B}$. Since $\boldsymbol{B}$ is positive definite, (iii) follows. ∎

## 2.2 Computation of the function value, the gradient vector and the Hessian matrix of $g(\cdot;\mu)$.

Let $(\boldsymbol{y};\mu) \in \mathrm{R}^m \times \mathrm{R}_{++}$. In order to compute the function value, the gradient vector and the Hessian matrix of $g(\boldsymbol{y};\mu)$, it suffices to solve the system of equations (8). In fact, if $(\boldsymbol{X}, y_{m+1}, \boldsymbol{S}) = (\boldsymbol{X}(\boldsymbol{y};\mu), y_{m+1}(\boldsymbol{y};\mu), \boldsymbol{S}(\boldsymbol{y};\mu)) \in \mathrm{S}^n_{++} \times \mathrm{R} \times \mathrm{S}^n_{++}$ is a solution of (8), then they are computed as

$$\begin{aligned}
g(\boldsymbol{y};\mu) &= -f(\boldsymbol{y},\boldsymbol{X}(\boldsymbol{y};\mu);\mu) \\
&= \left(\sum_{p=1}^{m}\boldsymbol{A}_p y_p - \boldsymbol{A}_0\right)\bullet \boldsymbol{X}(\boldsymbol{y};\mu) - \sum_{p=1}^{m} b_p y_p + \mu \log\det \boldsymbol{X}(\boldsymbol{y};\mu), \\
\boldsymbol{\nabla} g(\boldsymbol{y};\mu) &= (\boldsymbol{A}_1\bullet \boldsymbol{X}(\boldsymbol{y};\mu) - b_1,\ldots,\boldsymbol{A}_m\bullet \boldsymbol{X}(\boldsymbol{y};\mu) - b_m)^T \ \text{ (see (i) Lemma 2.1)}, \\
\boldsymbol{\nabla}^2 g(\boldsymbol{y};\mu) &= \left(\boldsymbol{C} - \boldsymbol{c}\boldsymbol{c}^T/B_{m+1,m+1}\right) \ \text{ (see (ii) Lemma 2.1)}.
\end{aligned}$$

Here $\boldsymbol{C} \in \mathrm{S}^m$, $\boldsymbol{c} \in \mathrm{R}^m$ and $B_{m+1,m+1} \in \mathrm{R}$ are given by (9) with $\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{y};\mu)$ and $\boldsymbol{S} = \boldsymbol{S}(\boldsymbol{y};\mu)$.

We can reduce the system of equations (8) to a single equation

$$\mu \boldsymbol{I} \bullet \left(\boldsymbol{A}_0 - \sum_{p=1}^{m}\boldsymbol{A}_p y_p - \boldsymbol{I} y_{m+1}\right)^{-1} = b_{m+1} \tag{12}$$

in the single variable $y_{m+1}$ with an additional positive definite condition

$$\left(\boldsymbol{A}_0 - \sum_{p=1}^{m}\boldsymbol{A}_p y_p - \boldsymbol{I} y_{m+1}\right) \in \mathrm{S}^n_{++}. \tag{13}$$

Then the other parts $\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{y};\mu)$ and $\boldsymbol{S} = \boldsymbol{S}(\boldsymbol{y};\mu)$ of the solution of (8) are computed by

$$\boldsymbol{S} \ = \ \left(\boldsymbol{A}_0 - \sum_{p=1}^{m}\boldsymbol{A}_p y_p - \boldsymbol{I} y_{m+1}\right) \ \text{and } \boldsymbol{X} = \mu \boldsymbol{S}^{-1}.$$

Let $\mu \in \mathrm{R}_{++}$ and $\boldsymbol{y} \in \mathrm{R}^m$ be fixed arbitrary. Define

$$\phi(y_{m+1}) = \mu \boldsymbol{I} \bullet \left(\boldsymbol{A}_0 - \sum_{p=1}^{m}\boldsymbol{A}_p y_p - \boldsymbol{I} y_{m+1}\right)^{-1}$$

for every $y_{m+1}$ satisfying (13). We want solve $\phi(y_{m+1}) = b_{m+1}$. Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of the matrix $\boldsymbol{A}_0 - \sum_{p=1}^{m}\boldsymbol{A}_p y_p$ involved in the definition of $\phi$. Let $\lambda_{\min}$ denote the minimum eigenvalue among $\lambda_1, \lambda_2, \ldots, \lambda_n$. Then the positive definite condition (13) turns out to be $y_{m+1} < \lambda_{\min}$, and

$$\left.\begin{aligned}
\phi(y_{m+1}) &= \mu \sum_{i=1}^{n} \frac{1}{\lambda_p - y_{m+1}}, \\
\frac{d\phi(y_{m+1})}{dy_{m+1}} &= \mu \sum_{i=1}^{n} \frac{1}{(\lambda_p - y_{m+1})^2} = \mu \boldsymbol{I} \bullet \left(\boldsymbol{A}_0 - \sum_{p=1}^{m}\boldsymbol{A}_p y_p - \boldsymbol{I} y_{m+1}\right)^{-2}.
\end{aligned}\right\} \tag{14}$$

We also see that $\phi$ is a strictly convex and strictly increasing function on $(-\infty, \lambda_{\min})$, and that it satisfies

$$\phi(y_{m+1}) \to 0 \text{ as } y_{m+1} \to -\infty \ \text{ and } \phi(y_{m+1}) \to +\infty \text{ as } y_{m+1} \to \lambda_{\min}.$$

Therefore the equation $\phi(y_{m+1}) = b_{m+1}$ has the unique solution $y^*_{m+1}$ in the interval

$$[\lambda_{\min} - n\mu/b_{m+1}, \lambda_{\min} - \mu/b_{m+1}].$$

To solve the equation $\phi(y_{m+1}) = b_{m+1}$, we could apply the Newton method. Because of the strict convexity, however, the generated sequence $\{y^k_{m+1} : k = 0, 1, 2 \dots\}$ would always overestimate the solution $y^*_{m+1}$; $y^*_{m+1} < y^k_{m+1}$ $(k = 1, 2, \dots)$. To avoid this disadvantage, we will propose a distinct method which is similar to the one often employed in the trust region method for nonlinear unconstrained optimization. Let $\bar{y}_{m+1} \in [\lambda_{\min} - n\mu/b_{m+1}, \lambda_{\min} - \mu/b_{m+1}]$ be a current iterate. We approximate the function $\phi$ by a function of the form

$$\psi(y_{m+1}) = \frac{\beta}{\alpha - y_{m+1}}.$$

Here the two real parameters $\alpha$ and $\beta$ are determined by

$$\bar{y}_{m+1} < \alpha, \ \ \psi(\bar{y}_{m+1}) = \phi(\bar{y}_{m+1}) \ \text{ and } \ \frac{d\psi(\bar{y}_{m+1})}{dy_{m+1}} = \frac{d\phi(\bar{y}_{m+1})}{dy_{m+1}}.$$

Then we choose the next iterate $y^+_{m+1}\alpha - \beta/b_{m+1}$ by solving the equation $\psi(y_{m+1}) = b_{m+1}$.

At each iteration of this algorithm, we need to evaluate

$$\phi(y_{m+1}) = \mu \boldsymbol{I} \bullet \boldsymbol{S}^{-1} \ \text{ and } \ \frac{d\phi(y_{m+1})}{dy_{m+1}} = \mu \boldsymbol{S}^{-1} \bullet \boldsymbol{S}^{-1},$$

where $\boldsymbol{S} = \boldsymbol{A}_0 - \sum_{p=1}^m \boldsymbol{A}_p y_p - \boldsymbol{I} y_{m+1}$. If the matrix $\boldsymbol{S}$ allows a sparse Cholesky factorization $\boldsymbol{S} = \boldsymbol{L}\boldsymbol{L}^T$, we can efficiently carried out the computation of both $\phi(y_{m+1})$ and $d\phi(y_{m+1})/dy_{m+1}$ by using $\boldsymbol{L}$. Or if we compute all the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ of the matrix $\boldsymbol{A}_0 - \sum_{p=1}^m \boldsymbol{A}_p y_p$ in advance, we can easily compute the value and the first derivative of the function $\phi$ using the relations (14). Usually the former method is cheaper than the latter unless the number of iterations gets larger.

## 2.3 Approximation of the central trajectory in the $\boldsymbol{y}$-space.

This section will provide the theoretical foundations of predictor procedures of path-following methods given in Section 3.

Let $\mu \in \mathrm{R}_{++}$. Assume that $\boldsymbol{y} = \boldsymbol{y}(\mu) \in \mathrm{R}^m$ is a (unique) minimizer of the function $g(\cdot;\mu)$ over $\mathrm{R}^m$. In view of Lemma 2.1, $\boldsymbol{y} = \boldsymbol{y}(\mu)$ satisfies $\boldsymbol{A}_p \bullet \boldsymbol{X}(\boldsymbol{y};\mu) = b_p$ $(p = 1, 2, \dots, m)$. Recall that $\boldsymbol{X}(\boldsymbol{y};\mu)$ is characterized by the condition (8). Therefore $\boldsymbol{y} = \boldsymbol{y}(\mu)$ satisfies

$$\left.\begin{array}{l} \boldsymbol{A}_p \bullet \boldsymbol{X} = b_p \ (p = 1, 2, \dots, m), \ \boldsymbol{I} \bullet \boldsymbol{X} = b_{m+1}, \\ \displaystyle\sum_{p=1}^m \boldsymbol{A}_p y_p + \boldsymbol{I} y_{m+1} + \boldsymbol{S} = \boldsymbol{A}_0, \\ \boldsymbol{X}\boldsymbol{S} = \mu \boldsymbol{I}, \ \boldsymbol{X} \in \mathrm{S}^n_{++}, \ \boldsymbol{S} \in \mathrm{S}^n_{++} \end{array}\right\} \qquad (15)$$

for some $(\boldsymbol{X}, y_{m+1}, \boldsymbol{S}) = (\boldsymbol{X}(\boldsymbol{y};\mu), y_{m+1}(\boldsymbol{y};\mu), \boldsymbol{S}(\boldsymbol{y};\mu)) \in \mathrm{S}^n \times \mathrm{R} \times \mathrm{S}^n$. This is exactly the condition that characterizes the central trajectory of the primal-dual SDPs (3) and (4).

Thus we may regard $\{(\boldsymbol{y}(\mu),\mu) : \mu \in \mathrm{R}_{++}\}$ as the projection of the central trajectory on the space $\mathrm{R}^m$ of the Lagrange multiplier vector $\boldsymbol{y}$. In the remainder of this section, we derive systems of linear equations that determine the first derivative $\dot{\boldsymbol{y}}(\mu)$ and the second derivative $\ddot{\boldsymbol{y}}(\mu)$. With these derivatives, we can approximate the trajectory $\{(\boldsymbol{y}(\mu),\mu) : \mu \in \mathrm{R}_{++}\}$ by either

$$\{((\boldsymbol{y}(\bar{\mu}) + \dot{\boldsymbol{y}}(\bar{\mu})(\mu - \bar{\mu}), \mu) : \mu \in \mathrm{R}_{++}\}$$

(the first order approximation) or

$$\left\{ \left( \boldsymbol{y}(\bar{\mu}) + \dot{\boldsymbol{y}}(\bar{\mu})(\mu - \bar{\mu}) + \frac{1}{2}\ddot{\boldsymbol{y}}(\bar{\mu})(\bar{\mu})(\mu - \bar{\mu})^2, \mu \right)) : \mu \in \mathrm{R}_{++} \right\}$$

(the second order approximation) in a neighborhood of each $\bar{\mu} \in \mathrm{R}_{++}$. These first and second order approximations of the trajectory will be used as a mathematical model for our predictor procedures presented in Section 3.

For simplicity of notation, we write

$$\boldsymbol{X}(\mu) = \boldsymbol{X}(\boldsymbol{y}(\mu);\mu), \;\; y_{m+1}(\mu) = y_{m+1}(\boldsymbol{y}(\mu);\mu) \;\; \text{and } \boldsymbol{S}(\mu) = \boldsymbol{S}(\boldsymbol{y}(\mu);\mu).$$

Then $(\boldsymbol{X}, \boldsymbol{y}, y_{m+1}, \boldsymbol{S}) = (\boldsymbol{X}(\mu), \boldsymbol{y}(\mu), y_{m+1}(\mu), \boldsymbol{S}(\mu))$ satisfies all the identities in (15). Differentiating those identities once and twice in $\mu$, we obtain

$$\left.\begin{array}{l} \boldsymbol{A}_p \bullet \dot{\boldsymbol{X}} = 0 \; (p = 1, 2, \ldots, m), \; \boldsymbol{I} \bullet \dot{\boldsymbol{X}} = 0, \\ \displaystyle\sum_{p=1}^{m} \boldsymbol{A}_p \dot{y}_p + \boldsymbol{I}\dot{y}_{m+1} + \dot{\boldsymbol{S}} = \boldsymbol{O}, \; \dot{\boldsymbol{X}}\boldsymbol{S} + \boldsymbol{X}\dot{\boldsymbol{S}} = \boldsymbol{I} \end{array}\right\}$$

and

$$\left.\begin{array}{l} \boldsymbol{A}_p \bullet \ddot{\boldsymbol{X}} = 0 \; (p = 1, 2, \ldots, m), \; \boldsymbol{I} \bullet \ddot{\boldsymbol{X}} = 0, \\ \displaystyle\sum_{p=1}^{m} \boldsymbol{A}_p \ddot{y}_p + \boldsymbol{I}\ddot{y}_{m+1} + \ddot{\boldsymbol{S}} = \boldsymbol{O}, \; \ddot{\boldsymbol{X}}\boldsymbol{S} + \boldsymbol{X}\ddot{\boldsymbol{S}} = -2\dot{\boldsymbol{X}}\dot{\boldsymbol{S}}, \end{array}\right\}$$

respectively. Here $(\dot{\boldsymbol{X}}, \dot{\boldsymbol{y}}, \dot{y}_{m+1}, \dot{\boldsymbol{S}}) = (\dot{\boldsymbol{X}}(\mu), \dot{\boldsymbol{y}}(\mu), \dot{y}_{m+1}(\mu), \dot{\boldsymbol{S}}(\mu))$ and $(\ddot{\boldsymbol{X}}, \ddot{\boldsymbol{y}}, \ddot{y}_{m+1}, \ddot{\boldsymbol{S}}) = (\ddot{\boldsymbol{X}}(\mu), \ddot{\boldsymbol{y}}(\mu), \ddot{y}_{m+1}(\mu), \ddot{\boldsymbol{S}}(\mu))$ denote the first and second derivatives of $(\boldsymbol{X}, \boldsymbol{y}, y_{m+1}, \boldsymbol{S}) = (\boldsymbol{X}(\mu), \boldsymbol{y}(\mu), y_{m+1}(\mu), \boldsymbol{S}(\mu))$, respectively. From these equations and (ii) of Lemma 2.1, we obtain

$$\left.\begin{array}{rcl} \boldsymbol{\nabla}^2 g(\boldsymbol{y};\mu)\dot{\boldsymbol{y}} &=& \dfrac{-\boldsymbol{b} + b_{m+1}\boldsymbol{c}/B_{m+1,m+1}}{\mu}, \\ \boldsymbol{\nabla}^2 g(\boldsymbol{y};\mu)\ddot{\boldsymbol{y}} &=& \frac{-\boldsymbol{r} + r_{m+1}\boldsymbol{c}/B_{m+1,m+1}}{\mu}, \\ \dot{y}_{m+1} &=& \dfrac{-b_{m+1}/\mu - \boldsymbol{c}^T\dot{\boldsymbol{y}}}{B_{m+1,m+1}}, \\ \dot{\boldsymbol{S}} &=& -\sum_{p=1}^{m} \boldsymbol{A}_p \dot{y}_p - \boldsymbol{I}\dot{y}_{m+1}, \\ \dot{\boldsymbol{X}} &=& \boldsymbol{S}^{-1} - \boldsymbol{X}\dot{\boldsymbol{S}}\boldsymbol{S}^{-1} = \dfrac{\boldsymbol{X} - \boldsymbol{X}\dot{\boldsymbol{S}}\boldsymbol{X}}{\mu}, \\ \ddot{y}_{m+1} &=& \dfrac{r_{m+1} - \boldsymbol{c}^T\ddot{\boldsymbol{y}}}{B_{m+1,m+1}}, \end{array}\right\} \quad (16)$$

where $B_{m+1,m+1}$ and $\boldsymbol{c}$ are given by (9), and $r_q$ $(q = 1, 2, \ldots, m+1)$ are given by

$$\left.\begin{array}{lcl} r_q & = & -2\boldsymbol{A}_q \bullet \dot{\boldsymbol{X}}\dot{\boldsymbol{S}}\boldsymbol{X} \ (q = 1, 2, \ldots, m), \\ r_{m+1} & = & -2\boldsymbol{I} \bullet \dot{\boldsymbol{X}}\dot{\boldsymbol{S}}\boldsymbol{X}. \end{array}\right\} \tag{17}$$

In view of (ii) of Lemma 2.1, we know that the common coefficient matrix $\boldsymbol{\nabla}^2 g(\boldsymbol{y};\mu) = \boldsymbol{C} - \boldsymbol{c}\boldsymbol{c}^T/B_{m+1,m+1}$ of the first and second equations in (16) is positive definite, so that we can utilize its Cholesky factorization to solve the equations.

# 3 Predictor-corrector path-following algorithms in the space $\mathrm{R}^m$ of the Lagrange multiplier vector $\boldsymbol{y}$.

## 3.1 A prototype algorithm using the Newton method.

Among various candidates for neighborhoods of the trajectory $\{(\boldsymbol{y}(\mu), \mu) \in \mathrm{R}^m \times \mathrm{R}_{++} : \mu \in \mathrm{R}_{++}\}$, we employ the one based on the self-concordant theory [23]. In the LP case, Kojima-Megiddo-Mizuno-Shindoh [16] showed the fact that $\{g(\cdot;\mu) : \mu \in \mathrm{R}_{++}\}$ forms a self-concordant family. Shida [26] extended this fact to a general class of linear optimization problems over closed convex cones which includes our SDP case. For every $\epsilon \in \mathrm{R}_{++}$, define

$$N(\epsilon) = \left\{(\boldsymbol{y};\mu) \in \mathrm{R}^m \times \mathrm{R}_{++} : \boldsymbol{\nabla} g(\boldsymbol{y};\mu)^T \boldsymbol{\nabla}^2 g(\boldsymbol{y};\mu)^{-1} \boldsymbol{\nabla} g(\boldsymbol{y};\mu) \leq \mu\epsilon\right\}. \tag{18}$$

**Algorithm 3.1.**

Step 0: Let $0 < \epsilon_c < \epsilon_p$, $0 < \gamma < 1$. Choose a $\mu^0 \in \mathrm{R}_{++}$ and a $\bar{\boldsymbol{y}}^0 \in \mathrm{R}^m$. Let $k = 0$.

Step 1-N: (Corrector procedure using the Newton method) Let $\boldsymbol{z} = \bar{\boldsymbol{y}}^k$. To approximately solve the problem of minimizing the strictly convex smooth function $g(\cdot;\mu^k)$ over $\mathrm{R}^m$, repeat the damped Newton iteration:

- solve $\boldsymbol{\nabla}^2 g(\boldsymbol{z};\mu^k)\boldsymbol{d} = -\boldsymbol{\nabla} g(\boldsymbol{z};\mu^k)$ in the search direction $\boldsymbol{d} \in \mathrm{R}^m$,
- choose a step length $\alpha \in (0, 1]$ such that $\boldsymbol{z}^+ = \boldsymbol{z} + \alpha\boldsymbol{d}$ and $g(\boldsymbol{z}^+;\mu^k) < g(\boldsymbol{z};\mu^k)$ (for example, use Armijo's or Wolfe's line search rule),
- replace $\boldsymbol{z}^+$ by $\boldsymbol{z}$

until $(\boldsymbol{z}, \mu^k) \in N(\epsilon_c)$. Let $\boldsymbol{y}^k = \boldsymbol{z}$. For the computation of the function value $g(\boldsymbol{z};\mu^k)$, the gradient vector $\boldsymbol{\nabla} g(\boldsymbol{z};\mu^k)$ and the Hessian matrix $\boldsymbol{\nabla}^2 g(\boldsymbol{z};\mu^k)$, see Section 2.2.

Step 2-N: (Predictor procedure using the Newton method) Let $(\boldsymbol{X}^k, y^k_{m+1}, \boldsymbol{S}^k) \in \mathrm{S}^n_{++} \times \mathrm{R} \times \mathrm{S}^n_{++}$ be the solution of (8) with $(\boldsymbol{y};\mu) = (\boldsymbol{y}^k, \mu^k) \in \mathrm{R}^m \times \mathrm{R}_{++}$. Compute an approximation $\dot{\boldsymbol{y}} = \dot{\boldsymbol{y}}^k \in \mathrm{R}^m$ of the first derivative $\dot{\boldsymbol{y}}(\mu^k)$ and an approximation $\ddot{\boldsymbol{y}} = \ddot{\boldsymbol{y}}^k \in \mathrm{R}^m$ of the second derivative $\ddot{\boldsymbol{y}}(\mu^k)$ by solving the systems of linear equations (16) with $(\boldsymbol{X}, \boldsymbol{y}, y_{m+1}, \boldsymbol{S}, \mu) = (\boldsymbol{X}^k, \boldsymbol{y}^k, y^k_{m+1}, \boldsymbol{S}^k, \mu^k)$. Let $\delta = 1$. Repeat

- $\delta = \gamma\delta$, $\bar{\mu} = (1-\delta)\mu^k$ and $\bar{\boldsymbol{y}} = \boldsymbol{y}^k + \dot{\boldsymbol{y}}^k(\bar{\mu} - \mu^k) + \ddot{\boldsymbol{y}}^k(\bar{\mu} - \mu^k)^2/2$.

until $(\bar{\boldsymbol{y}}, \bar{\mu}) \in N(\epsilon_p)$. Let $\mu^{k+1} = \bar{\mu}$ and $\bar{\boldsymbol{y}}^{k+1} = \bar{\boldsymbol{y}}$.

Step 3: Replace $k+1$ by $k$ and return to Step 1-N.

To design efficient algorithms for large scale SDPs based on the prototype algorithm described above, we need to incorporate various practical techniques which have been developed in the field of unconstrained optimization. In particular, we can employ the BFGS quasi-Newton method for the minimization of the function $g(\cdot;\mu^k)$ over the entire Euclidean space $\mathrm{R}^m$. This will be discussed in the next Section 3.2. Another important technique is an application of iterative methods such as the CG (conjugate gradient) method and the CR (conjugate residual) method for solving the system of equations (16) . This will be discussed in Section 3.3.

## 3.2 Corrector procedure using the quasi-Newton BFGS method.

In Step 0, we initialize $\boldsymbol{H} = \boldsymbol{I}$, which we will update to approximate the inverse of the Hessian matrix $\boldsymbol{\nabla}^2 g(\boldsymbol{z}, \mu^k)$ in Step 1-BFGS below.

Step 1-BFGS: (Corrector procedure using the BFGS quasi-Newton method) To approximately solve the problem of minimizing the strictly convex smooth function $g(\cdot;\mu^k)$ over $\mathrm{R}^m$, repeat the BFGS quasi-Newton iteration:

- $\boldsymbol{d} = -\boldsymbol{H}\boldsymbol{\nabla} g(\boldsymbol{z}, \mu^k)$ (note that $\boldsymbol{H}$ is corresponding to an approximation of $\boldsymbol{\nabla}^2 g(\boldsymbol{z}, \mu^k)$),
- choose a step length $\alpha \in (0,1]$ such that $\boldsymbol{z}^+ = \boldsymbol{z} + \alpha\boldsymbol{d}$ and $g(\boldsymbol{z}^+;\mu^k) < g(\boldsymbol{z};\mu^k)$ (for example, use Armijo's or Wolfe's line search rule),
- let $\boldsymbol{H}^+ = \boldsymbol{H} - \dfrac{\boldsymbol{H}\boldsymbol{\eta}\boldsymbol{\sigma}^T + \boldsymbol{\sigma}(\boldsymbol{H}\boldsymbol{\eta})^T}{\boldsymbol{\sigma}^T\boldsymbol{\eta}} + \left(1 + \dfrac{\boldsymbol{\eta}^T\boldsymbol{H}\boldsymbol{\eta}}{\boldsymbol{\sigma}^T\boldsymbol{\eta}}\right)\dfrac{\boldsymbol{\sigma}\boldsymbol{\sigma}^T}{\boldsymbol{\sigma}^T\boldsymbol{\eta}}$, where $\boldsymbol{\sigma} = \boldsymbol{z}^+ - \boldsymbol{z}$ and $\boldsymbol{\eta} = \boldsymbol{\nabla} g(\boldsymbol{z}^+;\mu^k) - \boldsymbol{\nabla} g(\boldsymbol{z};\mu^k)$ (the BFGS quasi-Newton update),
- replace $\boldsymbol{z}^+$ by $\boldsymbol{z}$ and $\boldsymbol{H}^+$ by $\boldsymbol{H}$

until

$$\boldsymbol{\nabla} g(\boldsymbol{z};\mu^k)^T \boldsymbol{H} \boldsymbol{\nabla} g(\boldsymbol{z};\mu^k) \leq \mu\epsilon_c. \tag{19}$$

Let $\boldsymbol{y}^k = \boldsymbol{z}$.

It should be noted that the stopping criteria $(\boldsymbol{z}, \mu^k) \in N(\epsilon_c)$, which also requires to solve $\boldsymbol{\nabla}^2 g(\boldsymbol{z};\mu^k)\boldsymbol{d} = \boldsymbol{\nabla} g(\boldsymbol{z};\mu^k)$ in $\boldsymbol{d}$, for the Newton iteration in Step-N of Algorithm 3.1 has been now replaced by (19).

## 3.3 Predictor procedure using preconditioned iterative methods.

To save the CPU time in the corrector procedure (Step 1-N or Step 1-BFGS), it is better to start from an initial point $\boldsymbol{z} = \bar{\boldsymbol{y}}^k$ which is sufficiently close to the minimizer $\boldsymbol{y}(\mu^k)$, so that the role of the predictor procedure becomes important. We could replace the second-order model $\bar{\boldsymbol{y}} = \boldsymbol{y}^k + \dot{\boldsymbol{y}}^k(\bar{\mu} - \mu^k) + \ddot{\boldsymbol{y}}^k(\bar{\mu} - \mu^k)^2/2$ for approximating the trajectory $\{(\boldsymbol{y}(\mu), \mu) : \mu \in \mathrm{R}_{++}\}$ by a simpler first-order model $\bar{\boldsymbol{y}} = \boldsymbol{y}^k + \dot{\boldsymbol{y}}^k(\bar{\mu} - \mu^k)$. However, we confirmed that the second-order model works more effectively than the first-order model through our preliminary numerical experiments.

In the predictor procedure, Step 2-N of Algorithm 3.1, we need to compute an approximation $\dot{\boldsymbol{y}} = \dot{\boldsymbol{y}}^k \in \mathrm{R}^m$ of the first derivative $\dot{\boldsymbol{y}}(\mu^k)$ and an approximation $\ddot{\boldsymbol{y}} = \ddot{\boldsymbol{y}}^k \in \mathrm{R}^m$ of the second derivative $\ddot{\boldsymbol{y}}(\mu^k)$ by solving the system of linear equations (16) with $(\boldsymbol{X}, \boldsymbol{y}, y_{m+1}, \boldsymbol{S}, \mu) = (\boldsymbol{X}^k, \boldsymbol{y}^k, y^k_{m+1}, \boldsymbol{S}^k, \mu^k)$. Since the common coefficient matrix $\boldsymbol{\nabla}^2 g(\boldsymbol{y}; \mu) = \boldsymbol{C} - \boldsymbol{c}\boldsymbol{c}^T/B_{m+1,m+1}$ of the first and the second equations of the system (16) is positive definite by (ii) of Lemma 2.1, we can utilize its Cholesky factorization to solve the first and the second equations exactly. But the coefficient matrix is fully dense in general, and solving the equations exactly by the Cholesky factorization gets more and more expensive when the dimension $m$ of the equations gets larger. We propose below to use iteration methods such as the CG (conjugate gradient) method and the CR (conjugate residual) method.

Let $\boldsymbol{H}^k$ denote the BFGS quasi-Newton matrix, which we have computed in the corrector procedure of the $k$th iteration. Let $\boldsymbol{L}^k(\boldsymbol{L}^k)^T$ be the Cholesky factorization of $\boldsymbol{H}^k$. Preconditioning the first equation of the system (16) by the matrices $\boldsymbol{L}^k$ and $(\boldsymbol{L}^k)^T$, we transform it into the equivalent system of equations

$$\boldsymbol{L}^k \boldsymbol{\nabla}^2 g(\boldsymbol{y}^k; \mu^k)(\boldsymbol{L}^k)^T \boldsymbol{u} = \boldsymbol{L}^k \left( \frac{\boldsymbol{b} + b_{m+1}\boldsymbol{c}/B_{m+1,m+1}}{\mu} \right) \quad \text{and } \dot{\boldsymbol{y}} = (\boldsymbol{L}^k)^T \boldsymbol{u}, \tag{20}$$

and then apply the CG method (or the CR method) to the resultant system. If the matrix $\boldsymbol{H}^k$ approximates $\boldsymbol{\nabla}^2 g(\boldsymbol{y}^k; \mu^k)$ in a reasonable accuracy, this preconditioning technique considerably improves the condition number of the original coefficient matrix $\boldsymbol{\nabla}^2 g(\boldsymbol{y}^k; \mu^k)$. It should be noted that we can carry out the actual computation in the CG method (or the CR method) without factorizing the matrix $\boldsymbol{H}^k$.

We can apply the CG method (or the CR method) with the same preconditioning matrix $\boldsymbol{H}^k$ to the second equation of the system (16).

## 3.4 Further sophisticated techniques for large scale SDPs.

When the dimension $m$ of the variable vector $\boldsymbol{z}$ is large, say, more than several thousand, it is usually impossible to store the entire $m \times m$ BFGS matrix $\boldsymbol{H}$ in standard workstations. In such cases, we may replace the full BFGS quasi-Newton update used in Step-BFGS by the limited memory BFGS update [24]. In their recent paper [21], Morales and Nocedal proposed to use the limited memory BFGS quasi-Newton matrix for preconditioning the CG method. In our predictor procedure described in the previous subsection, we can employ the limited memory BFGS quasi-Newton matrix for preconditioning the CG method (or the CR method).

As we have warned repeatedly, the approximation of the minimizer $\boldsymbol{y}(\mu)$ of $g(\cdot; \mu)$ over $\mathrm{R}^m$ in a higher accuracy is not an easy task at all because the condition number of the Hessian matrix $\boldsymbol{\nabla}^2 g(\cdot; \bar{\mu})$ of the objective function $g(\cdot; \mu)$ to be minimized over $\mathrm{R}^m$ gets worse rapidly when the barrier parameter $\mu$ gets smaller. In the prototype algorithm with the use of Newton method in Section 3.1 and also its variation with the use of the quasi-Newton BFGS method in Section 3.2, we have suggested to use Armijo's or Wolfe's rule for the line search along the search direction. But those techniques may not be powerful enough to compute an approximate minimizer with a high accuracy, and we may need to replace those line search techniques by a more sophisticated but more powerful technique such as the trust region method.

# 4 Preliminary numerical results.

The primal purpose here is to verify that our methods works numerically and to investigate the effectiveness of the corrector procedure using the BFGS quasi-Newton method and the predictor procedure using the CG (or the CR) method preconditioned by the BFGS quasi-Newton matrix. All the numerical results were done by MATLAB Version 5.2 in a Macintosh 9600.

## 4.1 Test problems.

We generated three types of SDPs, an SDP relaxation of a box constrained quadratic 0-1 program, a norm minimization problem, and a linear matrix inequality.

**Box constrained quadratic 0-1 program** (abbreviated by BQIP). Let $\boldsymbol{Q}$ be an $n \times n$ symmetric matrix whose components $Q_{ij} = Q_{ji}$ $(1 \leq i \leq j \leq n)$ are chosen from random numbers uniformly distributed in the interval $(0.0, 1.0)$. Consider the problem:

$$\begin{array}{ll} \text{Minimize} & \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} \\ \text{subject to} & x_i = -1 \;\text{ or } 1 \;(i = 1, 2, \ldots, n). \end{array}$$

We can compute a lower bound of the objective value by solving a SDP relaxation of this problem:

$$\begin{array}{ll} \text{Minimize} & \boldsymbol{Q} \bullet \boldsymbol{X} \\ \text{subject to} & X_{ii} = 1 \;(i = 1, 2, \ldots, n), \;\boldsymbol{I} \bullet \boldsymbol{X} \leq n + 1, \;\boldsymbol{X} \in \mathrm{S}^n_+. \end{array}$$

The inequality constraint $\boldsymbol{I} \bullet \boldsymbol{X} \leq n + 1$ is redundant, but is added to transform the SDP into our standard form SDP (3).

**Norm minimization problem** (abbreviated by NMIN). Let $\boldsymbol{F}_p$ $(p = 0, 1, 2, \ldots, m)$ be $q \times r$ real rectangular matrices. We consider the problem of minimizing $\left\| \boldsymbol{F}_0 - \sum_{p=1}^{m} y_p \boldsymbol{F}_p \right\|$ in the vector variable $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)^T \in \mathrm{R}^m$. Here $\|\boldsymbol{F}\|$ denotes the 2 norm of $\boldsymbol{F}$, *i.e.*, $\|\boldsymbol{F}\| = \max_{\|\boldsymbol{u}\|=1} \|\boldsymbol{F}\boldsymbol{u}\|$ = the square root of the maximum eigenvalue of $\boldsymbol{F}^T\boldsymbol{F}$. We can reformulate this problem as an SDP:

$$\left.\begin{array}{ll}\text{Maximize} & y_{m+1} \\ \text{subject to} & \displaystyle\sum_{p=1}^{m} \boldsymbol{A}_p y_p + \boldsymbol{I} y_{m+1} + \boldsymbol{S} = \boldsymbol{A}_0,\ \boldsymbol{S} \in \mathrm{S}^n_+.\end{array}\right\} \tag{21}$$

Here

$$\begin{array}{rcl} n & = & q+r,\ \boldsymbol{A}_0 = \begin{pmatrix} \boldsymbol{O} & \boldsymbol{F}_0^T \\ \boldsymbol{F}_0 & \boldsymbol{O} \end{pmatrix}, \\ \boldsymbol{A}_p & = & \begin{pmatrix} \boldsymbol{O} & -\boldsymbol{F}_p^T \\ -\boldsymbol{F}_p & \boldsymbol{O} \end{pmatrix} \ (p = 1, 2, \ldots, m). \end{array}$$

If we define

$$\left.\begin{array}{lcl} b_p & = & 0 \ (p = 1, 2, \ldots, m), \\ b_{m+1} & = & 1, \boldsymbol{b} = (b_1, b_2, \ldots, b_{m+1})^T \in \mathrm{R}^{m+1}, \end{array}\right\} \tag{22}$$

then the dual of the SDP above turns out to be our standard form SDP (3). In our numerical experiments, we assigned a random number uniformly distributed in the interval $(0.0, 1.0)$ to each component of $\boldsymbol{F}_p$ $(p = 0, 1, 2, \ldots, m)$.

**Linear matrix inequality** (abbreviated by LMI). Let $\boldsymbol{F}_p$ $(p = 0, 1, 2, \ldots, m)$ be $\ell \times \ell$ real symmetric matrices. Consider an LMI: $\boldsymbol{F}_0 - \sum_{p=1}^{m} \boldsymbol{F}_p y_p \in \mathrm{S}^\ell_+$.This LMI has a solution $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)^T \in \mathrm{R}^m$ if and only if the following SDP has a nonnegative optimal value.

$$\left.\begin{array}{ll}\text{Maximize} & y_{m+1} \\ \text{subject to} & \displaystyle\sum_{p=1}^{m} \boldsymbol{F}_p y_p + \boldsymbol{I} y_{m+1} + \boldsymbol{S} = \boldsymbol{F}_0,\ y_{m+1} \leq 1,\ \boldsymbol{S} \in \mathrm{S}^\ell_+.\end{array}\right\} \tag{23}$$

Let

$$\begin{array}{rcl} n & = & \ell+1,\ \boldsymbol{A}_0 = \begin{pmatrix} \boldsymbol{F}_0 & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{pmatrix} \in \mathrm{S}^n, \\ \boldsymbol{A}_p & = & \begin{pmatrix} \boldsymbol{F}_p & \boldsymbol{0} \\ \boldsymbol{0}^T & 0 \end{pmatrix} \in \mathrm{S}^n \ (p = 1, 2, \ldots, m). \end{array}$$

Then we can rewrite the SDP (23) as the SDP (21), which we have derived from the norm minimization problem above. Therefore if we define $\boldsymbol{b} = (b_1, b_1, \ldots, b_{m+1})^T \in \mathrm{R}^{m+1}$ as in (22), we obtain our standard form SDP (3) as its dual. In our numerical experiment, we assigned a random number uniformly distributed in the interval $(-1.0, 0.0)$ to each component of $\boldsymbol{F}_p$ $(p = 0, 1, 2, \ldots, m)$.

## 4.2 Preliminary numerical results.

We wrote a MATLAB code for four variants of our algorithms using either of the Newton method or the BFGS quasi-Newton method as a corrector procedure and using either of

| Corrector | Newton | Newton | BFGS | BFGS |
|---|---|---|---|---|
| Predictor | 1st-order | 2nd-order | 1st-oder | 2nd-oder |
| $\ell$ | 36 | 24 | 34 | 19 |
| $\mu^\ell$ | 5.68e-5 | 2.04e-4 | 2.27e-4 | 2.90e-4 |
| rel.error | 3.20e-6 | 1.20e-5 | 1.15e-5 | 1.70e-5 |
| p.f.error | 5.98e-5 | 1.97e-4 | 2.83e-4 | 8.07e-4 |
| min.eig. $\boldsymbol{X}^\ell$ | 1.38e-6 | 2.47e-6 | 5.50e-6 | 7.02e-6 |
| min.eig.$\boldsymbol{S}^\ell$ | 1.54e-6 | 2.75e-6 | 6.18e-6 | 7.89e-6 |
| max.eig.$\boldsymbol{X}^\ell\boldsymbol{S}^\ell$ | 5.68e-5 | 1.02e-4 | 2.27e-4 | 2.90e-4 |

Table 1: SDP relaxation of box constrained QOPs with $n = 100$.

the first-order model and the second-order model as a predictor procedure. We did not incorporate any sparsity technique into the code. We assigned the following values to the parameters and the initial points: $\epsilon_p = 0.05$, $\epsilon_c = 0.001$, $\gamma = 0.5$, $\mu^0 = \sqrt{m}$, $\boldsymbol{y}^0 = \boldsymbol{0} \in \mathrm{R}^m$, and used Armijo's rule for the line search along the search directions generated by either the Newton method or the quasi-Newton BFGS method.

Table 1 shows numerical results on the SDP relaxation of the BQIP with $n = 100$. $\ell$ denotes the number of iterations when the algorithm stopped because of a numerical difficulty in Step 1-N or Step 1-BFGS; we regarded that the algorithm had gotten into a numerical difficulty either when the number of iterations to determine a legitimate step length in Armijo's rule exceeded $m$ or when the number of iterations in Step 1-N or Step 1-BFGS to compute $\boldsymbol{y}^{k+1}$ from the initial point $\boldsymbol{y}^k$ exceeded $m$. Also the following symbols are used.

$$
\begin{aligned}
\text{rel.error} &= \text{the relative error } = \frac{\boldsymbol{A}_0 \bullet \boldsymbol{X}^\ell - \sum_{p=1}^{m+1} b_p y_p^\ell}{\max\{|\boldsymbol{A}_0 \bullet \boldsymbol{X}^\ell|,\ 1.0\}}. \\
\text{p.f.error} &= \text{the primal feasibility error } = \left(\sum_{p=1}^{m+1}\left(b_p - \boldsymbol{A}_p \bullet \boldsymbol{X}^\ell\right)^2\right)^{1/2}. \\
\text{min.eig.}\boldsymbol{X}^\ell &= \text{the minimum eigenvalue of } \boldsymbol{X}^\ell. \\
\text{min.eig.}\boldsymbol{S}^\ell &= \text{the minimum eigenvalue of } \boldsymbol{S}^\ell. \\
\text{max.eig.}\boldsymbol{X}^\ell\boldsymbol{S}^\ell &= \text{the maximum eigenvalue of } \boldsymbol{X}^\ell\boldsymbol{S}^\ell.
\end{aligned}
$$

From the numerical results in Table 1, we observe that all the variants could generate only low accuracy approximate optimal solutions; we need more sophisticated implementation to compute higher accuracy optimal solutions.

Table 2 shows how

- the condition number of $\boldsymbol{\nabla}^2 g(\boldsymbol{y}^k; \mu^k)$,
- the condition number of $\boldsymbol{L}^k\boldsymbol{\nabla}^2 g(\boldsymbol{y}^k; \mu^k)(\boldsymbol{L}^k)^T$, where $\boldsymbol{L}^k(\boldsymbol{L}^k)^T$ denotes the Cholesky factorization of the quasi-Newton BFGS matrix $\boldsymbol{H}^k$ (see Section 3.3), and
- # CG, the number of iterations in CG method in the predictor procedure

| $k$ | $\mu^k$ | condition number $\nabla^2 g(\boldsymbol{y}^k;\mu^k)$ | $\boldsymbol{L}^k\nabla^2 g(\boldsymbol{y}^k;\mu^k)(\boldsymbol{L}^k)^T$ | # CG |
|---|---|---|---|---|
| 1 | 1.00e+1 | 2.36e+2 | 2.12e+1 | 17 |
| 4 | 4.22e+0 | 6.04e+2 | 3.86e+1 | 25 |
| 7 | 1.19e+0 | 1.12e+3 | 4.71e+1 | 17 |
| 10 | 1.48e-1 | 1.58e+3 | 6.21e+1 | 18 |
| 13 | 1.85e-2 | 1.30e+4 | 6.32e+1 | 17 |
| 16 | 2.32e-3 | 1.04e+5 | 1.28e+2 | 21 |
| 17 | 1.16e-3 | 2.08e+5 | 1.19e+2 | 15 |
| 18 | 5.79e-4 | 4.16e+5 | 1.09e+2 | 13 |
| 19 | 2.90e-4 | 8.33e+5 | 1.55e+2 | - |
| | | | average #CG = 16.8 | |

Table 2: Change in the condition numbers of the Hessian matrix and the preconditioned Hessian matrix

changed along the sequence $\{(\boldsymbol{y}^k,\mu^k)\}$ generated the BFGS + 2nd-order version applied to the SDP relaxation of the BQIP with $n = 100$ (the last column of Table 1) . We see that the condition number of $\nabla^2 g(\boldsymbol{y}^k;\mu^k)$ had gotten worse rapidly as the iteration proceeds. But the condition number of preconditioned matrix $\boldsymbol{L}^k\nabla^2 g(\boldsymbol{y}^k;\mu^k)(\boldsymbol{L}^k)^T$ remained stable, and there was no increasing tendency in the number of iterations of the CG method in the predictor procedure; hence the preconditioning the Hessian matrix $\nabla^2 g(\boldsymbol{y}^k;\mu^k)$ by $\boldsymbol{H}^k = \boldsymbol{L}^k(\boldsymbol{L}^k)^T$ worked effectively. Note that the algorithm stopped at the 19th iteration because of a numerical difficulty in the corrector procedure before executing the predictor procedure.

In the rest of the numerical experiments which we will report, we stopped the algorithms when $\mu^k$ gets less than $10^{-4}$ or when the relative error gets less than $10^{-3}$. Table 3 shows numerical results on the SDP relaxation of the BQIP with $n = 25,\ 50,\ 100$. Here

$$\begin{aligned}
\#\text{it} &= \text{the total number of iterations,}\\
\#\text{Newton} &= \text{the total number of Newton iterations,}\\
\#\text{BFGS} &= \text{the total number of BFGS updates,}\\
\#\ \text{Chol} &= \text{the total number of Cholesky factorizations of}\\
&\phantom{=}\ \text{the dual matrix variable } \boldsymbol{S},\\
\#\text{CG} &= \text{the total number of CG iterations.}
\end{aligned}$$

We observe in Table 3 that the second-order model for approximating the trajectory in the $\boldsymbol{y}$-space works effectively in both of the Newton and the quasi-Newton BFGS variants.

Table 4 shows numerical results on NMIN with $q = 10$, $r = 15$, $m = 50, 100$ and LMI with $m = 50, 100$. In the dense case, one iteration of the Newton method, the quasi-Newton BFGS method and the CG method require $O(m^3)$, $O(m^2)$ and $O(m^2)$ arithmetic operations, respectively. Also each computation of the Hessian matrix $\nabla^2 g(\boldsymbol{y};\mu)$ requires $O(m^2n^2 + mn^3)$ arithmetic operations. Although #CG and #BFGS of the quasi-Newton BFGS variant are larger than #Newton of the Newton variant in all cases, the arithmetic operations required for the BFGS and the CG methods are much less than those required

| n | Corrector Predictor | Newton 1st-order | Newton 2nd-order | BFGS 1st-order | BFGS 2nd-order |
|---|---|---|---|---|---|
| 25 | #it | 21 | 13 | 21 | 13 |
| | #Newton | 75 | 39 | | |
| | #BFGS | | | 122 | 89 |
| | #CG | | | 178 | 117 |
| | #Chol | 268 | 148 | 621 | 470 |
| | CPU (sec.) | 7.72 | 4.63 | 4.81 | 3.54 |
| 50 | #it | 27 | 13 | 27 | 15 |
| | #Newton | 102 | 38 | | |
| | #BFGS | | | 177 | 143 |
| | #CG | | | 283 | 207 |
| | #Chol | 347 | 141 | 974 | 844 |
| | CPU (sec.) | 23.92 | 9.03 | 19.34 | 12.82 |
| 100 | #it | 24 | 21 | 27 | 17 |
| | #Newton | 89 | 70 | | |
| | #BFGS | | | 190 | 147 |
| | #CG | | | 369 | 303 |
| | #Chol | 371 | 262 | 1358 | 1138 |
| | CPU (sec.) | 222.69 | 171.75 | 135.75 | 99.06 |

Table 3: SDP relaxation of BQIP

for the Newton method. But we have to pay particular attention to a critical difference in the #Chol row which shows the total number of the Cholesky factorization of the dual matrix variable $\boldsymbol{S} = \boldsymbol{A}_0 - \sum_{p=1}^{m} \boldsymbol{A}_p y_p - \boldsymbol{I} y_{m+1}$ required in the Newton variant and the BFGS variant. In the dense case, one Cholesky factorization requires $O(n^3)$ arithmetic operations. When $n$ gets larger than $m$, the amount of work required in the Cholesky factorization can be the most time consuming part of the BFGS variant unless we exploit the aggregate sparsity of the data matrices $\boldsymbol{A}_0, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_m$. See [12].

# 5 Concluding remarks.

**(A)** Although we have reported some numerical results, we are not satisfied. The current code is written in pure MATLAB language, so that it is very slow. It does not take any sparsity consideration into account, so that it can not solve large scale problems. Also it encounters numerical instability when the barrier parameter gets smaller, and it could not generate highly accurate optimal solutions in our numerical experiments. There remains many issues to be studied further towards more practically efficient implementation for large scale problems. Among others, we need to explore the use of

- sparse Cholesky factorization of the dual matrix variable $\boldsymbol{S} = \boldsymbol{A}_0 - \sum_{p=1}^{m} \boldsymbol{A}_p y_p - \boldsymbol{I} y_{m+1}$,

| | | NMIN | | LMI | |
|---|---|---|---|---|---|
| | Corrector | Newton | BFGS | Newton | BFGS |
| m | Predictor | 2nd-order | 2nd-order | 2nd-order | 2nd-order |
| | #it | 23 | 23 | 21 | 21 |
| 50 | #Newton | 67 | | 68 | |
| | #BFGS | | 197 | | 212 |
| | #CG | | 220 | | 223 |
| | #Chol | 221 | 1115 | 224 | 1426 |
| | CPU (sec.) | 9.27 | 7.65 | 15.09 | 10.90 |
| | #it | 26 | 25 | 26 | 26 |
| 100 | #Newton | 79 | | 92 | |
| | #BFGS | | 294 | | 349 |
| | #CG | | 383 | | 435 |
| | #Chol | 231 | 1735 | 281 | 2901 |
| | CPU (sec.) | 38.12 | 26.34 | 48.93 | 35.68 |

Table 4: NMIN with $q = 10$, $r = 15$ and LMI

- the limited memory quasi-Newton BFGS method [21, 24].

**(B)** As we have mentioned several times so far, the main disadvantage of our method lies in the fact that the condition number of the Hessian matrix $\boldsymbol{\nabla}^2 g(\cdot;\mu)$ of the objective function $g(\cdot;\mu)$ to be minimized over $\mathrm{R}^m$ gets worse rapidly when the barrier parameter $\mu$ gets smaller. Because of this reason, the current primitive implementation of our method is not able to generate high accuracy approximate solutions of SDPs. To overcome this difficulty, we need to explore the use of

- more stable and robust numerical methods for ill-conditioned unconstrained minimization of $g(\cdot;\mu)$ over $\mathrm{R}^m$.

Another possibility of getting a higher accuracy approximate solution of an SDP is a switch to a standard primal-dual interior-point method. Recall that approximate solutions $\hat{\boldsymbol{X}}$ of the primal SDP (3) and $(\hat{\boldsymbol{y}}, \hat{y}_{m+1}, \hat{\boldsymbol{S}})$ of the dual SDP (4) obtained by our method lies approximately on the central trajectory; $\boldsymbol{X}$ approximately satisfies the equality constraint of (3), $\hat{\boldsymbol{X}} \in \mathrm{S}^n_{++}$, $(\hat{\boldsymbol{y}}, \hat{y}_{m+1}, \hat{\boldsymbol{S}})$ satisfies the equality constraint of (4), $\hat{\boldsymbol{S}} \in \mathrm{S}^n_{++}$ and $\hat{\boldsymbol{X}}\hat{\boldsymbol{S}} = \hat{\mu}\boldsymbol{I}$ for some small positive $\hat{\mu}$. Thus the point serve as a high quality initial point from which we can apply a primal-dual interior-point method.

**(C)** Consider the following LOP (linear optimization problem) and its dual.

$$\text{Primal LOP:} \quad \left.\begin{array}{ll} \text{minimize} & \langle \boldsymbol{a}_0, \boldsymbol{x} \rangle \\ \text{subject to} & \langle \boldsymbol{a}_p, \boldsymbol{x} \rangle = b_p \ (p = 1, 2, \ldots, m), \\ & \langle \boldsymbol{e}, \boldsymbol{x} \rangle = b_{m+1}, \ \boldsymbol{x} \in \mathcal{K}_+. \end{array}\right\} \tag{24}$$

$$\left.\begin{array}{lll} \text{Dual LOP:} & \text{maximize} & \sum_{p=1}^{m+1} b_p y_p \\ & \text{subject to} & \sum_{p=1}^{m} \boldsymbol{a}_p y_p + \boldsymbol{e} y_{m+1} + \boldsymbol{s} = \boldsymbol{a}_0, \ \ \boldsymbol{s} \in \mathcal{K}_+. \end{array}\right\} \tag{25}$$

Here $\boldsymbol{a}_p$ $(p = 0, 1, 2, \ldots, m)$ are given data vectors in $\mathrm{R}^\ell$, $b_p \in \mathrm{R}$ $(p = 1, 2, \ldots, m)$ and $0 < b_{m+1} \in \mathrm{R}$ are given real numbers, $\langle \boldsymbol{a}, \boldsymbol{x} \rangle$ denotes an inner product of $\mathrm{R}^\ell$, $\mathcal{K}_+$ a closed symmetric cone in $\mathrm{R}^\ell$, and $\boldsymbol{e}$ denotes the unit vector in the Euclidean Jordan Algebra associated with the closed symmetric cone $\mathcal{K}_+$. We recommend the readers to refer to the paper [9, 10] for definitions of some basic materials, such as the unit vector, the determinant of $\boldsymbol{x} \in \mathcal{K}_{++}$ and the bilinear operator, used in the Euclidean Jordan Algebra. The LOP covers not only the LP and the SDP but also the SOCP (second-order cone programming problem). In the case of SDPs, the inner product $\langle \boldsymbol{a}, \boldsymbol{x} \rangle$ is corresponding to $\boldsymbol{A} \bullet \boldsymbol{X}$ and the unit vector $\boldsymbol{e}$ to the identity matrix $\boldsymbol{I}$. We can extend our methods proposed for the primal-dual pair of SDPs (3) and (4) to more general primal-dual pair of LOPs (24) and (25) in a straightforward way. Specifically, the logarithmic barrier function $f$ defined in (7) and the associated problem (6) turn out to be

$$\begin{array}{rcl} f(\boldsymbol{y}, \boldsymbol{x}; \mu) & = & \left\langle \boldsymbol{a}_0 - \sum_{p=1}^{m} \boldsymbol{a}_p y_p, \boldsymbol{x} \right\rangle + \sum_{p=1}^{m} b_p y_p - \mu \log \det \boldsymbol{x} \\ & & \text{for every } (\boldsymbol{y}, \boldsymbol{x}, \mu) \in \mathrm{R}^m \times \mathcal{K}_{++} \times \mathrm{R}_{++} \end{array}$$

and

$$\left.\begin{array}{ll} \text{minimize} & f(\boldsymbol{y}, \boldsymbol{x}; \mu) \\ \text{subject to} & \boldsymbol{x} \in \Omega_{++} \equiv \{\boldsymbol{x} \in \mathcal{K}_{++} : \langle \boldsymbol{e}, \boldsymbol{x} \rangle = b_{m+1}\}, \end{array}\right\}$$

respectively. Here $\mathcal{K}_{++}$ denotes the interior of the symmetric cone in $\mathrm{R}^\ell$. As a condition which characterizes the unique optimal solution $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{y}; \mu)$, we obtain

$$\langle \boldsymbol{e}, \boldsymbol{x} \rangle = b_{m+1}, \ \ \boldsymbol{e} y_{m+1} + \boldsymbol{s} = \boldsymbol{a}_0 - \sum_{p=1}^{m} \boldsymbol{a}_p y_p, \ \ \boldsymbol{x} \circ \boldsymbol{s} = \mu \boldsymbol{e}, \ \ \boldsymbol{s} \in \mathcal{K}_{++},$$

which is corresponding to (8) in the SDP case. Here $\circ$ denotes the bilinear operator of the underling Euclidean Jordan algebra associated with the symmetric cone $\mathcal{K}_+$. We can define a family of strictly convex smooth functions $g(\cdot; \mu)$ $(\mu \in \mathrm{R}_{++})$ whose minimizers forms the central trajectory in the space $\mathrm{R}^m$ of the Lagrange multiplier vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)^T$:

$$\begin{array}{rcl} g(\boldsymbol{y}; \mu) & \equiv & -\min\{f(\boldsymbol{y}, \boldsymbol{x}; \mu) : \boldsymbol{x} \in \Omega_{++}\} = -f(\boldsymbol{y}, \boldsymbol{x}(\boldsymbol{y}; \mu)) \\ & & \text{for every } (\boldsymbol{y}; \mu) \in \mathrm{R}^m \times \mathrm{R}_{++}. \end{array}$$

Details are omitted here.


## Acknowledgment.

The authors would like to thank Professor Hiroshi Yabe for valuable discussions and suggestions on the quasi-Newton BFGS method. In particular, he brought the authors' attention to the recent paper [21] on the limited memory quasi-Newton BFGS method.

## References


[1] F. Alizadeh, J.-P. A. Haeberly and M. L. Overton, "Primal-dual interior point methods for semidefinite programming: Convergence rates, stability and numerical results," *SIAM J. Optim.* **8** (1998) 746–768.

[2] S. J. Benson, Y. Ye and X. Zhang, "Solving large-scale sparse semidefinite programs for combinatorial optimization," *SIAM J. Optim.* **10** (2000) 443–461.

[3] B. Borchers, "CSDP 2.3 user's guide," *Optim. Methods and Software* **11-12** (1999) 597-611.

[4] S. Burer, R. D. C. Monteiro and Y. Zhang, "Solving semidefinite programs via nonlinear programming part I: transformations and derivatives," TR99-17, Dept. of Comp. and Appl., Rice Univ., Rice Univ., Houston, Texas 77005, USA, 1999.

[5] S. Burer, R. D. C. Monteiro and Y. Zhang, "Solving semidefinite programs via nonlinear programming part II: Interior point methods for a subclass of SDPs," TR99-23, Dept. of Comp. and Appl., Rice Univ., Houston, Texas 77005, USA, 1999.

[6] C. Choi and Y. Ye, "Solving sparse semidefinite programs using the dual scaling algorithm with an iterative solver," Comb. Optim. Lab., Univ. of Iowa, 2000.

[7] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations,* (Prentice-Hall Inc, Englewood Cliffs, NJ, 1983).

[8] K. Fujisawa, K. Nakata and M Kojima, "SDPA(SemiDefinite Programming Algorithm) User's Mannual — Version 5.00," B-308, Dept. of Math. and Comp. Sci., Tokyo Inst. of Tech., Tokyo 152-8552, Japan. Available via anonymous ftp at `ftp://ftp.is.titech.ac.jp/pub/OpRes/software/SDPA`

[9] J. Faraut and A. Korány, *Analysis on Symmetric Cones,* (Oxford University Press, Oxford, UK, 1994).

[10] L. Faybusovich, "Linear systems in Jordan algebras and primal-dual interior-point algorithms," *J. of Comp. and Appl. Math.* **86** (1997) 149-175.

[11] K. Fujisawa, M. Kojima and K. Nakata, "Exploiting sparsity in primal-dual interior-point methods for semidefinite programming," *Math. Prog.* **79** (1997) 235–253.

[12] M. Fukuda, M. Kojima, K. Murota and K. Nakata, "Exploiting sparsity in semidefinite programming via matrix completion I: General framework," *SIAM J. Optim.* **11** (2000) 647-674.

[13] C. Helmberg and F. Rendl, "Solving quadratic $(0, 1)$-problems by semidefinite programming and cutting planes," *Math. Prog.* **82** (1998) 291–315.

[14] C. Helmberg and F. Rendl, " A spectral bundle method for semidefinite programming," ZIB Preprint SC-97-37, Konrad-Zuse-Zentrum fur Informationstechnik Berlin, Aug. 1997. Revised October 1997. To appear in *SIAM J. Optim.*.

[15] C. Helmberg, F. Rendl, R. J. Vanderbei and H. Wolkowicz, "An interior-point method for semidefinite programming," *SIAM J. Optim.* **6** (1996) 342–361.

[16] M. Kojima, N. Megiddo, S. Mizuno and S. Shindoh, "Horizontal and vertical decomposition in interior point methods for linear programs," Working Paper, Dept. of Inf.. Sci., Tokyo Inst. of Tech., Tokyo 152-8552, December 1993.

[17] M. Kojima, N. Megiddo, S. Mizuno and S. Shindoh, "Decomposition in interior-point methods," B-281, Dept. of Inf. Sci., Tokyo Inst. of Tech., Tokyo 152-8552, April 1994.

[18] M. Kojima, S. Shindoh and S. Hara, "Interior-point methods for the monotone semidefinite linear complementarity problems," *SIAM J. Optim.* **7** (1997) 86–125.

[19] C.-J. Lin and R. Saigal, "An incomplete Cholesky factorization for dense symmetric positive definite matrices," *BIT* **40** (2000) 536–558.

[20] R. D. C. Monteiro, "Primal-dual path-following algorithms for semidefinite programming," *SIAM J. Optim.* **7** (1997) 663–678.

[21] J. L. Morales and J. Nocedal, "Algorithm PREQN: Fortran 77 subroutines for preconditioning the conjugate gradient method," ECE Dept., Northwestern Univ., Evanston IL 60208, USA, March 2000.

[22] K. Nakata, K. Fujisawa and M. Kojima, "Using the conjugate gradient method in interior-point methods for semidefinite programs" (in Japanese), *Proc. Inst. of Stat. Math.* **46** (1998) 297–316.

[23] Yu. E. Nesterov and A. Nemirovski, *Interior-Point Polynomial Algorithms in Convex Programming* (SIAM, Philadelphia, PA, 1994).

[24] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Math. Comp.* **35** (1980) 773–782.

[25] S. H. Schimieta and F. Alizadeh, "Associative and Jordan algebra, and polynomial time interior point algorithms for symmetric cones," RRR 12-99, Rutcor, Rutgers Univ., NJ 08854-8003, August 1999.

[26] M. Shida, "An interior-point smoothing techinique for Lagrange relaxation in convex programming," Working paper, Dept. of Mathematics, Kanagawa University 3-27-1, Rokkakubashi, Kanagawa-ku, Yokohama 221-8686, Japan, April 1998.

[27] J. F. Sturm, "Using SeDuMi 1.02, MATLAB toolbox for optimization over symmetric cones," *Optim. Methods and Software* **11–12** (1999) 625–653.

[28] M. J. Todd, K. C. Toh and R. H. Tütüncü, "On the Nesterov-Todd direction in semidefinite programming," *SIAM J. Optim.* **8** (1998) 769–796

[29] K. C. Toh and M. Kojima, "Solving some large scale semidefinite programs via the conjugate residual method," Working paper, Dept. of Math., National Univ. of Singapore, Singapore 119620, Singapore, August 2000.

[30] K. C. Toh, M. J. Todd and Tütüncü, "SDPT3 — a MATLAB software package for semidefinite programming, version 1.3," *Optim. Methods and Software* **11–12** (1999) 545–581.

[31] G. Zhao, "Interior point methods with decomposition for solving large scale linear programs," *Journal of Optimization Theory and Applications* **102** (1999) 169–192.

[32] G. Zhao, "A Log-barrier method with Benders decomposition for solving two-stage stochastic programs," Research Report No. 711, Dept of Mathematics, National University of Singapore, 1997. Revised 1999.

[33] G. Zhao, "A Lagrangian dual method with self-concordant barrier for multi-stage stochastic convex nonlinear programming," Working paper, Dept of Mathematics, National University of Singapore, 1998, revised 2000.

[34] R. J. Vanderbei and H. Yurttan Benson, "On formulating semidefinite programming problems as smooth convex nonlinear optimization problems," ORFE 99-01, Dept. of Oper. Res. and Financial Eng., Princeton Univ., Princeton NJ, 1999. Revised January 2000.