Research Reports on Information Sciences
Series B : Operations Research

A Primitive Interior Point Algorithm for Semidefinite Programs in *Mathematica*

Masakazu Kojima[†]

Classification: Mathematical Programming
December 1994, B-293

Department of Mathematical and Computing Sciences
Tokyo Institute of Technology
2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152, Japan
e-mail : kojima@is.titech.ac.jp

B-293 A Primitive Interior Point Algorithm for Semidefinite Programs in *Mathematica*, Masakazu Kojima†, December 1994

Abstract. In these few years, SDPs (semidefinite programs) have been studied intensively in mathematical programming circles. Particularly SDPs have important applications to combinatorial optimization and systems and control theory. The References include a partial list of literatures on applications of SDPs to these fields and computational methods for SDPs. Recently Kojima, Shindoh and Hara proposed an interior-point algorithm for monotone linear complementarity problems in symmetric matrices. When their algorithm is applied to SDPs, it works as a primal-dual interior-point algorithm which simultaneously generates approximate optimal solutions of a given SDP and its dual. The aim of this working paper is to present a simple and basic version of their primal-dual interior-point algorithm for SDPs together with a computer program of the algorithm in *Mathematica*, placing the emphasis on easy understanding of the fundamental idea of the algorithm without going into its theoretical details.

Section 1 describes a primal-dual pair of SDPs, and Section 2 describes the existence of the central trajectory which plays an essential role in the primal-dual interior-point algorithm for SDPs. Sections 3 and 4 are devoted to a Primitive INterior Point ALgorithm for SDPs. Although the PINPAL in *Mathematica* can solve only small dimensional SDPs, it will provide useful information on how we develop practical and/or more efficient computer programs of the primal-dual interior-point algorithm for SDPs. "PINPAL in *Mathematica* for Macintosh" and "PINPAL in *Mathematica* for Windows" are available via anonymous ftp into ftp.is.titech.ac.jp. The pinpal package is in "tar"ed and "compress"ed format with the name pinpal.tar.Z in the directory pub/OpRes/software.

Key words Semidefinite Programming, Primal-Dual Interior Point Method, *Mathematica*

† Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152, Japan
e-mail : kojima@is.titech.ac.jp

1. Semidefinite Programs.

Let $\mathcal{M}_n$ denote the set of $n \times n$ matrices, and $\mathcal{M}_n^{sym} \subset \mathcal{M}_n$ the set of $n \times n$ symmetric matrices. When $\boldsymbol{A} \in \mathcal{M}_n^{sym}$ is positive semi-definite (or positive definite), we write $\boldsymbol{A} \succeq \boldsymbol{O}$ (or $\boldsymbol{A} \succ \boldsymbol{O}$, respectively). Given $\boldsymbol{F}_j \in \mathcal{M}_n^{sym}$ $(j = 0, 1, 2, \ldots, m)$ and $c_j \in I\!R$ $(j = 1, 2, \ldots, m)$, a primal SDP $\mathcal{P}$ and its dual $\mathcal{D}$ are defined as

$$\begin{array}{rcll} \mathcal{P} & : & \text{minimize} & \sum_{j=1}^m c_j x_j \\ & & \text{subject to} & \boldsymbol{X} = \sum_{j=1}^m \boldsymbol{F}_j x_j - \boldsymbol{F}_0 \succeq \boldsymbol{O}. \\ \mathcal{D} & : & \text{maximize} & \boldsymbol{F}_0 \bullet \boldsymbol{Y} \\ & & \text{subject to} & \boldsymbol{F}_j \bullet \boldsymbol{Y} = c_j \ (j = 1, 2, \ldots, m), \ \boldsymbol{Y} \succeq \boldsymbol{O}. \end{array}$$

Here $\boldsymbol{A} \bullet \boldsymbol{B}$ denotes the inner product of matrices $\boldsymbol{A}$, $\boldsymbol{B} \in \mathcal{M}_n$, *i.e.*, $\boldsymbol{A} \bullet \boldsymbol{B} = \mathrm{Tr}\, \boldsymbol{A}^T \boldsymbol{B}$. If $(\boldsymbol{x}, \boldsymbol{X})$ satisfies the constraint $\boldsymbol{X} = \sum_{j=1}^m \boldsymbol{F}_j x_j - \boldsymbol{F}_0 \succeq \boldsymbol{O}$ of $\mathcal{P}$ (or if $\boldsymbol{Y}$ satisfies the constraint $\boldsymbol{F}_j \bullet \boldsymbol{Y} = c_j$ $(j = 1, 2, \ldots, m)$, $\boldsymbol{Y} \succeq \boldsymbol{O}$ of $\mathcal{D}$), $(\boldsymbol{x}, \boldsymbol{X})$ is a *feasible solution* of $\mathcal{P}$ (or $\boldsymbol{Y}$ is a *feasible solution* of $\mathcal{D}$, respectively). If in addition $(\boldsymbol{x}, \boldsymbol{X})$ satisfies $\boldsymbol{X} \succ \boldsymbol{O}$ (or if $\boldsymbol{Y}$ satisfies $\boldsymbol{Y} \succ \boldsymbol{O}$), then $(\boldsymbol{x}, \boldsymbol{X})$ is an *interior feasible solution* of $\mathcal{P}$ (or $\boldsymbol{Y}$ is an *interior feasible solution* of $\mathcal{D}$, respectively). For simplicity, we call $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y})$ a feasible (interior feasible or *optimal*) solution of the SDP if $(\boldsymbol{x}, \boldsymbol{X})$ and $\boldsymbol{Y}$ are feasible (interior feasible or *optimal*) solutions of $\mathcal{P}$ and $\mathcal{D}$. Throughout the paper we assume:

Assumption 1.1.

(i) There is an interior feasible solution $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}})$ of the SDP;

$$\tilde{\boldsymbol{X}} = \sum_{j=1}^m \boldsymbol{F}_j \tilde{x}_j - \boldsymbol{F}_0 \succ \boldsymbol{O}, \ \boldsymbol{F}_j \bullet \tilde{\boldsymbol{Y}} = c_j \ (j = 1, 2, \ldots, m), \ \tilde{\boldsymbol{Y}} \succ \boldsymbol{O}.$$

(ii) The set of matrices $\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_m \in \mathcal{M}_n^{sym}$ is linearly independent, *i.e.*, the system of homogeneous linear equations $\sum_{j=1}^m \boldsymbol{F}_j x_j = \boldsymbol{O}$ in m variables $x_1, x_2, \ldots, x_m$ has no nontrivial solution.

2. Optimality and the Central Trajectory.

Under Condition 1.1:

- The SDP has an optimal solution.
- $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y})$ is an optimal solution of the SDP if and only if

$$\boldsymbol{X} = \sum_{j=1}^m \boldsymbol{F}_j x_j - \boldsymbol{F}_0 \succeq \boldsymbol{O} \ (\textit{primal feasibility}), \tag{1}$$

$$\boldsymbol{F}_j \bullet \boldsymbol{Y} = c_j \ (j = 1, 2, \ldots, m), \ \boldsymbol{Y} \succeq \boldsymbol{O} \ (\textit{dual feasibility}), \tag{2}$$

$$\boldsymbol{X}\boldsymbol{Y} = \boldsymbol{O} \ (\textit{complementarity}). \tag{3}$$

For every positive semi-definite $\boldsymbol{X}$, $\boldsymbol{Y} \in \mathcal{M}_n^{sym}$, $\boldsymbol{X}\boldsymbol{Y} = \boldsymbol{O}$ if and only if $\boldsymbol{X} \bullet \boldsymbol{Y} = 0$. Hence if $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y})$ satisfies the primal feasibility (1) and the dual feasibility (2), the complementarity

$\boldsymbol{XY} = \boldsymbol{O}$ is equivalent to $\boldsymbol{X} \bullet \boldsymbol{Y} = 0$. As in the usual linear program in the Euclidean space, every feasible solution $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y})$ of the SDP satisfies that

$$0 \leq \boldsymbol{X} \bullet \boldsymbol{Y} = \sum_{j=1}^{m} c_j x_j - \boldsymbol{F}_0 \bullet \boldsymbol{Y}.$$

Hence the optimality conditions (1), (2) and (3) can be restated as "a feasible solution $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y})$ of the SDP is optimal if and only if the primal objective value $\sum_{j=1}^{m} c_j x_j$ coincides with the dual objective value $\boldsymbol{F}_0 \bullet \boldsymbol{Y}$." Also if $\boldsymbol{X},\ \boldsymbol{Y} \in \mathcal{M}_n^{sym}$ are positive semi-definite, all the eigenvalues of the matrix $\boldsymbol{XY}$ are real and nonnegative and their sum coincides with $\boldsymbol{X} \bullet \boldsymbol{Y}$. In Section 4, we utilize the average $\mu = \boldsymbol{X} \bullet \boldsymbol{Y}/n$ of the eigenvalues of $\boldsymbol{XY}$ as a measure of the complementarity.

Among the set of optimality conditions (1), (2) and (3), we relax the complementarity (3) as follows:

$$\left.\begin{array}{l} \boldsymbol{X} = \sum_{j=1}^{m} \boldsymbol{F}_j x_j - \boldsymbol{F}_0 \succ \boldsymbol{O}, \\ \boldsymbol{F}_j \bullet \boldsymbol{Y} = c_j \ (j = 1, 2, \ldots, m),\ \boldsymbol{Y} \succ \boldsymbol{O}, \\ \boldsymbol{XY} = \mu \boldsymbol{I} \ \text{ for } \exists \mu > 0. \end{array}\right\} \tag{4}$$

Then

- For every $\mu > 0$, there is a unique $(\boldsymbol{x}(\mu), \boldsymbol{X}(\mu), \boldsymbol{Y}(\mu))$ satisfying the condition (4).
- The set
$$\Gamma = \{(\boldsymbol{x}(\mu), \boldsymbol{X}(\mu), \boldsymbol{Y}(\mu)) : \mu > 0\}$$
forms a smooth curve running through the set of interior feasible solutions of the SDP, and converges to an optimal solution of the SDP as $\mu \to 0$.

We call Γ the *central trajectory*, the *central path* or the *path of centers*. Primal-dual interior-point methods developed for SDPs numerically trace the central trajectory Γ. See *e.g.* [1, 2, 3, 7, 11, 12].

3. A Primitive Interior Point Algorithm.

3.1. Outline.

We first prepare an initial point $(\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0)$ such that

$$\boldsymbol{X}^0 \succ \boldsymbol{O},\ \boldsymbol{Y}^0 \succ \boldsymbol{O}, \tag{5}$$

and compute the residual

$$\boldsymbol{P}^0 = \sum_{j=1}^{m} \boldsymbol{F}_j x_j^0 - \boldsymbol{F}_0 - \boldsymbol{X}^0,\ d_j^0 = c_j - \boldsymbol{F}_j \bullet \boldsymbol{Y}^0 \ (j = 1, 2, \ldots, m) \tag{6}$$

in the equality constraints of the SDPs $\mathcal{P}$ and $\mathcal{D}$ at the initial point $(\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0)$. Let $\xi \geq 1$ be a fixed number. It is convenient for our succeeding discussions to introduce the set

$$\Omega = \left\{ (\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}) : \begin{array}{c} \boldsymbol{X} \succ \boldsymbol{O},\ \sum_{j=1}^{m} \boldsymbol{F}_j x_j - \boldsymbol{F}_0 - \boldsymbol{X} = \theta \boldsymbol{P}^0, \\ \boldsymbol{Y} \succ \boldsymbol{O},\ c_j - \boldsymbol{F}_j \bullet \boldsymbol{Y} = \theta^0 d_j^0 \ (j = 1, 2, \ldots, m), \\ \theta^p (\boldsymbol{X}^0 \boldsymbol{Y}^0)/n \leq \xi (\boldsymbol{X}^p \bullet \boldsymbol{Y}^p)/n \ \text{ for } \exists \theta \in [0, 1] \end{array} \right\}.$$

The PINPAL, the Primitive INterior Point ALgorithm generates a sequence $\{(\boldsymbol{x}^p, \boldsymbol{X}^p, \boldsymbol{Y}^p)\}$ in Ω. Let $\mu^p = \boldsymbol{X}^p \bullet \boldsymbol{Y}^p/n$. Then we can rewrite the relation $(\boldsymbol{x}^p, \boldsymbol{X}^p, \boldsymbol{Y}^p) \in \Omega$ as

$$\boldsymbol{X}^p \succ \boldsymbol{O},\ \sum_{j=1}^{m} \boldsymbol{F}_j x_j^p - \boldsymbol{F}_0 - \boldsymbol{X}^p = \theta^p \boldsymbol{P}^0, \tag{7}$$

$$\boldsymbol{Y}^p \succ \boldsymbol{O},\ c_j - \boldsymbol{F}_j \bullet \boldsymbol{Y}^p = \theta^p d_j^0\ (j = 1, 2, \ldots, m), \tag{8}$$

$$\theta^p \mu^0 \leq \xi \mu^p \tag{9}$$

for some sequence $\{\theta^p : p = 0, 1, 2, \ldots, \} \subset [0, 1]$. Furthermore the sequence $\{\theta^p : p = 0, 1, 2, \ldots, \}$ satisfies

$$\left.\begin{array}{l} \theta^0 = \left\{ \begin{array}{ll} 0 & \text{if } (\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0) \text{ is feasible, } i.e., \ \boldsymbol{P}^0 = \boldsymbol{O} \text{ and } \boldsymbol{d}^0 = \boldsymbol{0}, \\ 1 & \text{otherwise }, \end{array} \right. \\ \theta^0 \geq \theta^p \geq \theta^{p+1} \geq 0\ (p = 1, 2, \ldots\ ). \end{array}\right\} \tag{10}$$

The properties (7) and (8) imply that the residuals in the equality constraints of the SDPs $\mathcal{P}$ and $\mathcal{D}$ decrease linearly at a common rate in each iteration. The θ^p appeared on the right hand side of (7) and (8) serves as a measure of the feasibility; the iterate $(\boldsymbol{x}^p, \boldsymbol{X}^p, \boldsymbol{Y}^p)$ is feasible if and only if $\theta^p = 0$. The property (10) ensures that once the iterate $(\boldsymbol{x}^p, \boldsymbol{X}^p, \boldsymbol{Y}^p)$ gets feasible then the feasibility is maintained from then on. On the other hand, the $\mu^p = \boldsymbol{X}^p \bullet \boldsymbol{Y}^p/n$ appeared on the right hand side of the property (9) serves as a measure of the complementarity. We see from (9) that if μ^p converges to 0 as $p \to +\infty$ then θ^p converges to 0 as $p \to +\infty$; hence $(\boldsymbol{x}^p, \boldsymbol{X}^p, \boldsymbol{Y}^p)$ with every sufficiently large p gives an approximate optimal solution of the SDP. Given a sufficiently small number $\epsilon^* > 0$, we will stop the iteration when

$$\left.\begin{array}{l} \| \sum_{j=1}^{m} \boldsymbol{F}_j x_j^p - \boldsymbol{F}_0 - \boldsymbol{X}^p \| = \theta^p \| \boldsymbol{P}^0 \| \leq \epsilon^*, \\ \left(\sum_{j=1}^{m} (c_j - \boldsymbol{F}_j \bullet \boldsymbol{Y}^p)^2 \right)^{1/2} = \theta^p \left(\sum_{j=1}^{m} (d_j^0)^2 \right)^{1/2} \leq \epsilon^*, \\ \mu^p = \dfrac{\boldsymbol{X}^p \bullet \boldsymbol{Y}^p}{n} \leq \epsilon^*. \end{array}\right\} \tag{11}$$

Here $\|\boldsymbol{A}\| = (\boldsymbol{A} \bullet \boldsymbol{A})^{1/2}$ for every $\boldsymbol{A} \in \mathcal{M}_n$. It should be noted that the inequality (9) prevents the complementarity $\mu^p = \boldsymbol{X}^p \bullet \boldsymbol{Y}^p/n$ converging to 0 before the iterates $(\boldsymbol{x}^p, \boldsymbol{X}^p, \boldsymbol{Y}^p)$ gets feasible (or θ^p attains 0).

As in many optimization methods, we compute a search direction $(\boldsymbol{u}, \boldsymbol{U}, \boldsymbol{V})$ at each iteration and then choose a step length α^p to generate a new iterate such that

$$(\boldsymbol{x}^{p+1}, \boldsymbol{X}^{p+1}, \boldsymbol{Y}^{p+1}) = (\boldsymbol{x}^p, \boldsymbol{X}^p, \boldsymbol{Y}^p) + \alpha^p (\boldsymbol{u}, \boldsymbol{U}, \boldsymbol{V}).$$

PINPAL (Primitive INterior Point ALgorithm).

Step 0: Choose an initial point and several parameters as follows:

- An initial point $(\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0)$ satisfying (5); for example, $(\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0) = (\boldsymbol{0}, \boldsymbol{I}, \boldsymbol{I})$.
- The maximum number p^* of iterations; for example, $p^* = 30$.
- An accuracy $\epsilon^* > 0$; for example $\epsilon^* = 10^{-6}$.
- Parameters $\beta^* \in [0, 1)$ and $\xi^* \geq 1$ which control search directions; for example, $\beta^* = 0.01$, $\xi^* = 2.0$.

- Parameters $\gamma^* \in (0,1)$ and $\alpha^* \in (0,1)$ which control step lengths; for example, $\gamma^* = 0.9$ and $\alpha^* = 0.7$.

Step 1: Initialization: Compute the residuals $\boldsymbol{P}^0$ and $\boldsymbol{d}^0$ in the equality constraints of the SDPs $\mathcal{P}$ and $\mathcal{D}$ at the initial point $(\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0)$ by (6). Let

$$\begin{aligned} \mu^0 &= \frac{\boldsymbol{X}^0 \bullet \boldsymbol{Y}^0}{n}, \\ \theta^0 &= \begin{cases} 0 & \text{if } (\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0) \text{ is feasible, } i.e., \ \boldsymbol{P}^0 = \boldsymbol{O} \text{ and } \boldsymbol{d}^0 = \boldsymbol{0}, \\ 1 & \text{otherwise .} \end{cases} \end{aligned}$$

Let $p = 0$.

Step 2: Stop the iteration if (11) holds or the iteration p exceeds p^*.

Step 3: Choose a search direction parameter β^p according to the rule (17).

Step 4: Let $\boldsymbol{R} = \beta^p \mu^p \boldsymbol{I} - \boldsymbol{X}^p \boldsymbol{Y}^p$. Solve the system of equations (14), (15) and (16) to compute a search direction $(\boldsymbol{u}, \boldsymbol{U}, \boldsymbol{V})$.

Step 5: Choose a step length α^p according to the rule (25).

Step 6: Generate a new iterate

$$(\boldsymbol{x}^{p+1}, \boldsymbol{X}^{p+1}, \boldsymbol{Y}^{p+1}) = (\boldsymbol{x}^p, \boldsymbol{X}^p, \boldsymbol{Y}^p) + \alpha^p (\boldsymbol{u}, \boldsymbol{U}, \boldsymbol{V}).$$

Step 7: Let

$$\begin{aligned} \mu^{p+1} &= \frac{\boldsymbol{X}^{p+1} \bullet \boldsymbol{Y}^{p+1}}{n}, \\ \theta^{p+1} &= (1 - \alpha^p)\theta^p. \end{aligned}$$

Step 8: Replace $p+1$ by p and go to Step 2.

We will explain more details of Steps 3, 4, 5 and 6 in the succeeding discussions.

3.2. Computing a Search Direction.

Assume that we have obtained the pth iterate $(\boldsymbol{x}^p, \boldsymbol{X}^p, \boldsymbol{Y}^p)$ satisfying the relations (7), (8) and (9). In Step 3, we choose a *search direction parameter* $\beta^p \in [0,1]$, which determines a target point $(\boldsymbol{x}', \boldsymbol{X}', \boldsymbol{y}') = (\boldsymbol{x}(\beta^p\mu^p), \boldsymbol{X}(\beta^p\mu^p), \boldsymbol{Y}(\beta^p\mu^p))$ on the central trajectory Γ (or an optimal solution $(\boldsymbol{x}', \boldsymbol{X}', \boldsymbol{y}')$ of the SDP if $\beta^p = 0$);

$$\left.\begin{array}{l} \boldsymbol{X}' = \sum_{j=1}^{m} \boldsymbol{F}_j x'_j - \boldsymbol{F}_0 \succ \boldsymbol{O}, \\ \boldsymbol{F}_j \bullet \boldsymbol{Y}' = c_j \ (j = 1, 2, \ldots, m), \ \boldsymbol{Y}' \succ \boldsymbol{O}, \\ \boldsymbol{X}'\boldsymbol{Y}' = \beta^p \mu^p \boldsymbol{I}. \end{array}\right\} \tag{12}$$

The search direction

$$(\boldsymbol{u}, \boldsymbol{U}, \boldsymbol{V}) \in I\!R^m \times \mathcal{M}_n^{sym} \times \mathcal{M}_n^{sym}$$

is defined as "a Newton direction" at $(\boldsymbol{x}^p, \boldsymbol{X}^p, \boldsymbol{Y}^p)$ for approximating the target point $(\boldsymbol{x}', \boldsymbol{X}', \boldsymbol{Y}')$ on the central trajectory;

$$\left.\begin{array}{l} \boldsymbol{U} - \sum_{j=1}^{m} \boldsymbol{F}_j u_j = \theta^p \boldsymbol{P}^0, \\ \boldsymbol{F}_i \bullet (\boldsymbol{V} + \boldsymbol{W}) = \theta^p d_i^0 \ (i = 1, 2, \ldots, m), \\ \boldsymbol{X}^p(\boldsymbol{V} + \boldsymbol{W}) + \boldsymbol{U}\boldsymbol{Y}^p = \boldsymbol{R} \equiv \beta^p \mu^p \boldsymbol{I} - \boldsymbol{X}^p \boldsymbol{Y}^p, \\ \boldsymbol{U} \in \mathcal{M}_n^{sym}, \ \boldsymbol{u} \in I\!R^m, \ \boldsymbol{V} \in \mathcal{M}_n^{sym}, \ \boldsymbol{W} \in \mathcal{M}_n^{skew}. \end{array}\right\} \tag{13}$$

Here $\mathcal{M}_n^{skew}$ denotes the set of $n \times n$ skew symmetric matrices and $\boldsymbol{W} \in \mathcal{M}_n^{skew}$ an auxiliary variable matrix. If we derived a standard Newton equation from (12) in a usual way, we would have the system of (13) without the auxiliary variable matrix $\boldsymbol{W}$. Unfortunately, however, the standard Newton equation derived from (4) does not necessarily have a solution. Our modified Newton equation (13) with the auxiliary variable matrix $\boldsymbol{W} \in \mathcal{M}_n^{skew}$ always has a unique solution. See [7, 8].

Transforming the system of equations (13), we obtain:

$$\sum_{j=1}^{m} a_{ij} u_j = b_i \ (i = 1, 2, \ldots, m), \tag{14}$$

$$\boldsymbol{U} = \sum_{j=1}^{m} \boldsymbol{F}_j u_j + \theta^p \boldsymbol{P}^0, \tag{15}$$

$$\boldsymbol{V} = \frac{(\boldsymbol{X}^p)^{-1}(\boldsymbol{R} - \boldsymbol{U}\boldsymbol{Y}^p) + \left((\boldsymbol{X}^p)^{-1}(\boldsymbol{R} - \boldsymbol{U}\boldsymbol{Y}^p)\right)^T}{2}, \tag{16}$$

where

$$\begin{aligned} a_{ij} &= \left(\boldsymbol{F}_i(\boldsymbol{X}^p)^{-1}\right) \bullet (\boldsymbol{Y}^p \boldsymbol{F}_j) \ \ (i, j = 1, 2, \ldots, m), \\ b_i &= \left(\boldsymbol{F}_i(\boldsymbol{X}^p)^{-1}\right) \bullet (\boldsymbol{R}^T - \theta^p \boldsymbol{Y}^p \boldsymbol{P}^0) - \theta^p d_i^0 \ (i = 1, 2, \ldots, m). \end{aligned}$$

The system of equation (14) in the scalar variables $u_1, u_2, \ldots, u_m$ is a standard system of m linear equations in m scalar variables. Under (ii) of Condition 1.1 the $m \times m$ coefficient matrix $\boldsymbol{A} = [a_{ij}]$ turns out to be nonsingular, so that (14) above has a unique solution $\boldsymbol{u} = (u_1, u_2, \ldots, u_m)^T$. Other parts $\boldsymbol{U}$ and $\boldsymbol{V}$ of the search direction can be easily computed by (15) and (16).

How we choose a search direction parameter $\beta^p \in [0, 1]$ greatly influences the computational efficiency. One way of choosing a search direction parameter β^p is to fix it to a sufficiently small β^* (for example, $\beta^* = 0.01$) throughout the iterations. If in addition we can take a long step length α^p (for example, $\alpha^p \geq 0.9$) at every iteration, we obtain an approximate optimal solution with a fine accuracy in a small number of iterations. A small search direction parameter β^p and a long step length α^p at an iteration, however, sometimes place a new iterate $(\boldsymbol{x}^{p+1}, \boldsymbol{X}^{p+1}, \boldsymbol{y}^{p+1})$ far from the central trajectory or make some of the nonnegative eigenvalues of $\boldsymbol{X}^{p+1}\boldsymbol{Y}^{p+1}$ extremely small compared with the average $\mu^{p+1} = \boldsymbol{X}^{p+1} \bullet \boldsymbol{Y}^{p+1}/n$ of the eigenvalues of $\boldsymbol{X}^{p+1}\boldsymbol{Y}^{p+1}$. Such a situation may not allow us to take a long step length in the succeeding iterations unless we bring the iterate back near to the central trajectory by choosing a larger search direction parameter. In the PINPAL (the Primitive INterior Point ALgorithm) presented in Section 3.1, we have employed the following rule of choosing a search direction parameter:

$$\beta^p = \begin{cases} 1 & \text{if } p = 0 \ \text{ or } \alpha^{p-1} < \alpha^*, \\ \beta^* & \text{otherwise.} \end{cases} \tag{17}$$

Here $\beta^* \in (0, 1)$ and $\alpha^* \in (0, 1)$ are parameters chosen in Step 0 (for example, $\beta^* = 0.01, \ \alpha^* = 0.7$).

3.3. Computing a Step Length.

We want to choose a step length $\alpha = \alpha^p > 0$ such that

$$(\boldsymbol{x}^{p+1}, \boldsymbol{X}^{p+1}, \boldsymbol{Y}^{p+1}) = (\boldsymbol{x}^p, \boldsymbol{X}^p, \boldsymbol{Y}^p) + \alpha(\boldsymbol{u}, \boldsymbol{U}, \boldsymbol{V}) \in \Omega. \tag{18}$$

First we observe that any step length $\alpha > 0$ satisfies

$$\begin{aligned}
&\sum_{j=1}^{m} \boldsymbol{F}_j(x_j^p + \alpha u_j) - \boldsymbol{F}_0 - (\boldsymbol{X}^p + \alpha \boldsymbol{U}) = (1-\alpha)\theta^p \boldsymbol{P}^0, \\
&c_j - \boldsymbol{F}_j \bullet (\boldsymbol{Y}^p + \alpha \boldsymbol{V}) = (1-\alpha)\theta^p d_j^0 \ (j = 1, 2, \ldots, m), \\
&\frac{(\boldsymbol{X}^p + \alpha \boldsymbol{U}) \bullet (\boldsymbol{Y}^p + \alpha \boldsymbol{V})}{n} = (1 - (1-\beta^p)\alpha)\mu^p + \frac{\alpha^2 \boldsymbol{U} \bullet \boldsymbol{V}}{n}.
\end{aligned}$$

Taking

$$\theta = (1 - \alpha^p)\theta^p,$$

we then have that

$$\begin{aligned}
&\sum_{j=1}^{m} \boldsymbol{F}_j(x_j^p + \alpha u_j) - \boldsymbol{F}_0 - (\boldsymbol{X}^p + \alpha \boldsymbol{U}) = \theta^{p+1} \boldsymbol{P}^0, \\
&c_j - \boldsymbol{F}_j \bullet (\boldsymbol{Y}^p + \alpha \boldsymbol{V}) = \theta^{p+1} d_j^0 \ (j = 1, 2, \ldots, m).
\end{aligned}$$

Therefore the relation (18) holds if and only if

$$\boldsymbol{X}^p + \alpha \boldsymbol{U} \succ \boldsymbol{O}, \tag{19}$$

$$\boldsymbol{Y}^p + \alpha \boldsymbol{V} \succ \boldsymbol{O}, \tag{20}$$

$$(1-\alpha)\theta^p \mu^0 \leq \xi \left((1 - (1-\beta^p)\alpha)\mu^p + \frac{\alpha^2 \boldsymbol{U} \bullet \boldsymbol{V}}{n} \right). \tag{21}$$

Let λ_{min} denote the minimum of all the eigenvalues of $(\boldsymbol{X}^p)^{-1}\boldsymbol{U}$ and $(\boldsymbol{Y}^p)^{-1}\boldsymbol{V}$. Then a step length α satisfies (19) and (20) if and only if

$$\alpha < \bar{\alpha} \equiv \begin{cases} +\infty & \text{if } \lambda_{min} \geq 0, \\ -1/\lambda_{min} & \text{if } \lambda_{min} < 0. \end{cases} \tag{22}$$

On the other hand, we can rewrite (21) as the quadratic inequality

$$0 \leq a\alpha^2 + b\alpha + c,$$

where

$$a = \frac{\xi \boldsymbol{U} \bullet \boldsymbol{V}}{n}, \ b = \theta^p \mu^0 - \xi(1-\beta^p)\mu^p, \ c = \xi\mu^p - \theta^p \mu^0.$$

It follows from the inequalities (9) and $\beta^p > 0$ that

$$c \geq 0 \quad \text{and} \quad b = -c + \xi\beta^p\mu^p > 0 \text{ if } c = 0.$$

We also know that if $\theta^p = 0$ then

$$a = 0, \ b = -\xi(1-\beta^p) \leq 0 \ \text{ and } c = \xi\mu^p.$$

Hence if we define

$$\hat{\alpha} \equiv \begin{cases} \dfrac{1}{1-\beta^p} & \text{if } \theta^p = 0 \text{ and } \beta^p < 1, \\ \dfrac{-b-\sqrt{b^2-4ac}}{2a} & \text{if } \theta^p > 0,\ a < 0 \text{ and } b^2 - 4ac \geq 0, \\ -\dfrac{c}{b} & \text{if } \theta^p > 0,\ a = 0 \text{ and } b < 0, \\ \dfrac{-b-\sqrt{b^2-4ac}}{2a} & \text{if } \theta^p > 0,\ a > 0,\ b^2 - 4ac \geq 0 \text{ and } b < 0, \\ +\infty & \text{otherwise,} \end{cases} \tag{23}$$

then $\hat{\alpha} > 0$ and

$$0 \leq a\alpha^2 + b\alpha + c \quad \text{if} \quad \alpha \in [0, \hat{\alpha}].$$

Consequently, we obtain that a step length α satisfies (19), (20) and (21) if and only if

$$0 \leq \alpha < \bar{\alpha},\ \alpha \leq \hat{\alpha}. \tag{24}$$

In the PINPAL presented in Section 3.1, we have employed the following rule of choosing a step length:

$$\alpha^p = \alpha = \min\{\gamma^* \bar{\alpha},\ \hat{\alpha},\ 1.0\}. \tag{25}$$

where $\gamma^* \in (0, 1)$ is a constant (for example, $\gamma^* = 0.9$).

4. A Program of the PINPAL in *Mathematica*.

4.1. The Organization.

The program consists of :

(a) Main procedure (see Section 4.3) :

- PINPAL.

(b) Sub procedures used in the main procedure PINPAL (see Section 4.4) :

- innerProduct.
- matrixNorm.
- vectorNorm.
- matrixTrace.
- getSDPdata.
- checkSDPdata.
- getResidual.
- getnewtonDirection.
- getAlphaBar.
- getStepLength.
- printParameters.
- printIteration.
- printSolution.

(c) Parameters necessary to specify when we run the main procedure PINPAL :

- problemFile = the name of a data file which includes data on an SDP (see (d) below and Section 4.2).
- pStar = p^*.
- epsilonStar = ϵ^*.
- betaStar = β^*.
- gammaStar = γ^*.
- alphaStar = α^*.
- xiStar = ξ^*.

(d) A data file which includes the following data of an SDP:

"Title and Comments"
m
n
$\boldsymbol{c}$
$\boldsymbol{F}_0$
$\boldsymbol{F}_1$
.
$\boldsymbol{F}_m$

Here we can write any comments between two quotation marks " " in the first line. The description of the vector $\boldsymbol{c}$ and the matrices $\boldsymbol{F}_i$ $(i = 1, 2, \ldots, m)$ needs to follow the list format of *Mathematica*. See the example below.

4.2. Example.

Consider an SDP $\mathcal{P}$ with

$$m = 3,\ n = 2,\ \boldsymbol{c} = (48, -8, 20)^T,$$
$$\boldsymbol{F}_0 = \begin{pmatrix} -11 & 5 \\ 5 & 23 \end{pmatrix},\ \boldsymbol{F}_1 = \begin{pmatrix} 10 & 4 \\ 4 & 14 \end{pmatrix},\ \boldsymbol{F}_2 = \begin{pmatrix} 4 & 7 \\ 7 & -8 \end{pmatrix},\ \boldsymbol{F}_3 = \begin{pmatrix} 12 & -8 \\ -8 & -2 \end{pmatrix}.$$

We put these data into a file with the name "example.dat" as follows:

```
"Example: m = 3, n=2, feasible"
3
2
{48, -8, 20}
{{-11, 5}, {5, 23}}
{{10, 4}, {4, 14}}
{{4, 7}, {7, -8}}
{{12, -8}, {-8, -2}}
```

Assigning appropriate values to the parameters listed in (c) of Section 4.1, we execute the main procedure as follows:

```
problemFile = "example.dat";
pStar = 30;
epsilonStar = 1.0 10^-6;
betaStar = 0.01;
xiStar = 2.0;
gammaStar = 0.9;
alphaStar = 0.7;
PINPAL[problemFile,pStar,epsilonStar,betaStar,xiStar,gammaStar,alphaStar];
```

Then we obtain the following output.

```
* Primitive INterior Point Algorithm *
* Masakazu Kojima, Tokyo Institute of Technology *
* Version 1.0, November 29, 1994 *
Example: m = 3, n=2, feasible
* Parameters *
pStar = 30
                   -6
epsilonStar = 1. 10
betaStar = 0.01
gammaStar = 0.9
alphaStar = 0.7
xiStar = 2.
* iteration = 0 *
mu = 1.  theta = 1.
beta = 1.  alpha  = 0.9
* iteration = 1 *
mu = 0.19  theta = 0.1
beta = 0.01  alpha  = 0.8632
* iteration = 2 *
mu = 0.0198632  theta = 0.01368
beta = 0.01  alpha  = 0.90281
* iteration = 3 *
mu = 0.00199867  theta = 0.00132955
beta = 0.01  alpha  = 0.90848
* iteration = 4 *
mu = 0.000199988  theta = 0.00012168
beta = 0.01  alpha  = 0.909035
* iteration = 5 *
mu = 0.0000199999  theta = 0.0000110686
beta = 0.01  alpha  = 0.909086
* iteration = 6 *
          -6                    -6
mu = 2. 10    theta = 1.00629 10
beta = 0.01  alpha  = 0.90909
* iteration = 7 *
          -7                    -8
mu = 2. 10    theta = 9.14816 10
beta = 0.01  alpha  = 0.909091
```

```
**** Computational Results *****
Total No. of Iteration = 8
                                      -7
Primal feasibility error = 2.24083 10
                                      -7
Dual   feasibility error = 2.18774 10
         -8
mu = 2. 10
                                   -8
Minimum Eigenvalue of xMat = 1. 10
Minimum Eigenvalue of yMat = 2.
                                        -8
Maximum Eigenvalue of xMat.yMat = 2. 10
** Approximate Optimal Solution **
Primal Objective Value = 24.
Dual   Objective Value = 24.
xVect = {0.816401, -1.14224, -1.21626}
               -8            -22              -22       -8
xMat  = {{1. 10  , -5.4625 10   }, {-5.4625 10   , 1. 10  }}
                         -14              -14
yMat  = {{2., 1.43417 10   }, {1.43417 10   , 2.}}
```

4.3. Main Procedure.

```
(* Primitive INterior Point ALgorithm *)
(* Version 1.0, November 29, 1994 *)
(* Masakazu Kojima *)
(* Dept. of Mathematical and Computing Sciences *)
(* Tokyo Institute of Technology *)
(* Oh-Okayama, Meguro, Tokyo 152, Japan *)
(* e-mail : kojima@is.titech.ac.jp *)

(* *)
Off[General::spell];
Off[General::spell1];
(* *)

(* Main procedure *)
PINPAL[problemFile_,pStar_,epsilonStar_,betaStar_,
  xiStar_,gammaStar_,alphaStar_] :=
Module[
  {problem, (* Working area for receiving data *)
            (* from problemFile *)
   titleAndComment, (* Title and comment stated in problemFile *)
   m,  (* Dimension of xVect *)
   n,  (* Size of matrices fMat[0],fMat[1],...,fMat[m] *)
   fMat,  (* fMat[0],fMat[1],...,fMat[m]; n x n matrices in SDP *)
   cVect, (* Primal cost vector; m-dim vector in SDP *)
   xVect, (* Primal variable vector x; m-dim vector *)
```

```
 xMat,  (* Primal variable matrix X; n x n matrix *)
 yMat,  (* Dual variable matrix; n x n matrix *)
 newtonTriplet, (* Working area for Newton directions *)
 uVect, (* Newton direction ---> xVect+alpha uVect; n-dim vector *)
 uMat,  (* Newton direction ---> xMat +alpha uMat; n x n matrix *)
 vMat,  (* Newton direction ---> xMat +alpha uMat; n x n matrix *)
 residualList, (* Working area for the residuals below *)
 p0Mat, (* Residual in the primal equality constraint *)
        (* at the initial point; n x n matrix *)
 normOFp0Mat, (* Matrix norm of p0Mat *)
 d0Vect,(* Residual in the dual equality constraint *)
        (* at the initial point; m-dim vector *)
 normOFd0Vect, (*Norm of d0Vect *)
 rMat,  (* Residual in the centering equation; n x n matrix *)
 mu,  (* Average complementarity = xMat \bullet yMat / n *)
 mu0, (* Average complementarity at the initial point *)
 pIteration,
 theta, (* Measure for feasibility *)
 alpha, (* Step length *)
 beta,  (* Search direction parameter at each iteration *)
 nonzero, (* A sufficiently small positive number *)
 i},
nonzero = 1.0 10^-10;
(* Reading SDP *)
problem = getSDPdata[problemFile];
titleAndComment = problem[[1]];
m = problem[[2]]; n = problem[[3]];
cVect = problem[[4]];
Array[fMat,m,0]; Do[fMat[i] = problem[[5+i]],{i,0,m}];
(* Printing Parameters *)
printParameters[titleAndComment,pStar,epsilonStar,
  betaStar,xiStar,gammaStar,alphaStar];
(* Checking SDP data *)
If[checkSDPdata[m,n,cVect,fMat] == False,Abort[]];
(* Initialization of Global variable matrices and vectors *);
xVect = Table[0.0,{m}]; xMat = IdentityMatrix[n];
yMat = IdentityMatrix[n];
uVect = Table[0.0,{m}]; uMat = Table[0.0,{n},{n}];
vMat = Table[0.0,{n},{n}];
p0Mat = Table[0.0,{n},{n}]; d0Vect = Table[0.0,{m}];
rMat = Table[0.0,{n},{n}];
(** Step 0 **) (* Initial points *)
xVect = Table[0.0,{m}]; xMat = IdentityMatrix[n];
yMat = IdentityMatrix[n];
(** Step 1 **) (* Initialization *)
pIteration = 0;
residualList = getResidual[m,n,cVect,fMat,xVect,xMat,yMat];
p0Mat  = residualList[[1]]; normOFp0Mat = matrixNorm[n,p0Mat];
d0Vect = residualList[[2]]; normOFd0Vect = vectorNorm[m,d0Vect];
```

```
  mu0 = innerProduct[n,xMat,yMat] / n; mu = mu0;
  If[(normOFp0Mat<nonzero)&&(normOFd0Vect<nonzero),
    theta=0.0,theta=1.0];
  While[
    (** Step 2 **)
    (    (   (theta normOFp0Mat  > epsilonStar)
           ||(theta normOFd0Vect > epsilonStar)
           ||(mu > epsilonStar)
         )
      && (pIteration < pStar)
    ),
    (** Step 3 **)
    If[((pIteration == 0)||(alpha < alphaStar)),
       beta = 1.0, beta = betaStar];
    (** Step 4 **)
    rMat = beta mu IdentityMatrix[n] - xMat.yMat;
    newtonTriplet = getNewtonDirection[m,n,cVect,fMat,xVect,
      xMat,yMat,theta p0Mat,theta d0Vect,rMat,beta,mu];
    uVect = newtonTriplet[[1]]; uMat = newtonTriplet[[2]];
    vMat  = newtonTriplet[[3]];
    (** Step 5 **)
    alpha = getStepLength[n,xMat,uMat,yMat,vMat,
              gammaStar,xiStar,theta,mu0,mu];
    printIteration[pIteration,theta,mu,alpha,beta];
    (** Step 6 **)
    xVect = xVect + alpha uVect; xMat = xMat + alpha uMat;
    yMat  = yMat  + alpha vMat;
    (** Step 7 **)
    mu=innerProduct[n,xMat,yMat]/n; theta=(1.0-alpha)theta;
    (** Step 8 *)
    pIteration = pIteration + 1;
  ];
  printSolution[m,n,cVect,fMat,pIteration,
    xVect,xMat,yMat,mu,theta,epsilonStar];
];
```

4.4. Sub Procedures.

```
(* Sub procedures used in PINPAL *)

innerProduct[n_,aMat_,bMat_] :=
Module[{i,j,sum},
  sum = 0.0; Do[sum=sum+aMat[[i,j]]bMat[[j,i]],{i,1,n},{j,1,n}];
  Return[sum]
];

matrixNorm[n_,aMat_] :=
Module[{i,j,sum},
  sum=0.0; Do[sum=sum+aMat[[i,j]]aMat[[j,i]],{i,1,n},{j,1,n}];
```

```
  sum=Sqrt[sum]; Return[sum]
];

vectorNorm[m_,dVect_] :=
Module[{i,sum},
  sum=0.0; Do[sum=sum+dVect[[i]]dVect[[i]],{i,1,m}];
  sum=Sqrt[sum]; Return[sum]
];

matrixTrace[n_,aMat_] :=
Module[{i,sum},
  sum=0.0; Do[sum=sum+aMat[[i,i]],{i,1,n}]; Return[sum]
];

(** Reading Problem Data **)
getSDPdata[problemFile_] :=
Module[{file,problem},
  file = OpenRead[problemFile];
  problem = ReadList[file,Expression];
  Close[file]; Return[problem];
];

(** Checking Problem Date **)
checkSDPdata[m_,n_,cVect_,fMat_] :=
Module[
  {i,dimOFcVect,dimOFfMat,noError},
  noError = True;
  (* Checking the dimension of cVect *)
  dimOFcVect = Dimensions[cVect];
  If[Dimensions[dimOFcVect][[1]] != 1,
    If[noError == True,Print["!! Error in the SDP data !!"]];
    noError = False; Print["cVect is not a vector!"];
    Print["cVect = ",cVect],
    If[dimOFcVect[[1]] != m,
      If[noError == True,Print["!! Error in the SDP data !!"]];
      noError = False; Print["cVect is not ",m," dimensional!"];
      Print["cVect = ",cVect]
    ]
  ];
  (* Checking the dimension of fMat[i] (i=0,1,...,m) *)
  Do[
    dimOFfMat = Dimensions[fMat[i]];
    If[Dimensions[dimOFfMat][[1]] != 2,
      If[noError == True,Print["!! Error in the SDP data !!"]];
      noError = False; Print["fMat[",i,"] is not a matrix!"];
      Print["fMat[",i,"] = ",fMat[i]],
      If[dimOFfMat[[1]] != n || dimOFfMat[[2]] != n,
        If[noError == True,Print["!! Error in the SDP data !!"]];
        noError = False;
```

```
        Print["fMat[",i,"] is not ",n," x ",n," !"];
        Print["fMat[",i,"] = ",fMat[i]]
      ]
    ];
    Clear[dimOFfMat],
    {i,0,m}
  ];
  If[noError == False,Return[False]];
  (* Checking the symmetry of fMat[i] *)
  Do[
    Do[
      If[fMat[i][[j,k]] != fMat[i][[k,j]],
        If[noError == True,Print["!! Error in the SDP data !!"]];
        noError = False;
        Print[fMat[i][[j,k]],
              " = fMat[",i,"][",j,",",k,"] != ",
                 "fMat[",i,"][",k,",",j,"] = ",
              fMat[i][[k,j]]," !"];
        Print["fMat[",i,"] = ",fMat[i]];
      ],{j,1,n},{k,j+1,n}
    ],{i,0,m}
  ];
  Return[noError];
];

(** ---> Step 1 **)
getResidual[m_,n_,cVect_,fMat_,xVect_,xMat_,yMat_] :=
Module[{pMat,dVect,j},
  pMat=-fMat[0]-xMat;
  Do[pMat=pMat+xVect[[j]]fMat[j],{j,1,m}];
  dVect=Table[cVect[[j]]-innerProduct[n,fMat[j],yMat],{j,1,m}];
  Return[{pMat,dVect}];
];

(** ---> Step 4 **)
getNewtonDirection[m_,n_,cVect_,fMat_,xVect_,xMat_,yMat_,
  pMat_,dVect_,rMat_,beta_,mu_]:=
Module[
  {uMat,uVect,vMat,
   vPwMat, (* = vMat + wMat *)
   gMat,   (* gMat[i] = fMat[i].xMat^{-1} (i=1,2,...,m) *)
   hMat,   (* hMat[j] = fMat[j].yMat      (j=1,2,...,m) *)
   aMat,   (* The coefficient matric of the system of *)
           (* linear equations in u_1,u_2,...,u_m *)
   bVect,  (* The right hand side of the system of *)
           (* linear equations in u_1,u_2,...,u_m *)
   iMat,   (* = rMat - pMat.yMat *)
   invxMat,(* = xMat^{-1} *)
   i,j,k},
```

```
  Array[gMat,m,0]; Array[hMat,m,0];
  aMat = Table[0.0,{m},{m}]; bVect = Table[0.0,{m}];
  invxMat = Inverse[xMat];
  Do[gMat[i] = fMat[i].invxMat,{i,1,m}];
  Do[hMat[j] = yMat.fMat[j],{j,1,m}];
  Do[aMat[[i,j]]=innerProduct[n,gMat[i],hMat[j]],{i,1,m},{j,1,m}];
  iMat=Transpose[rMat] - yMat.pMat;
  Do[bVect[[i]]=innerProduct[n,gMat[i],iMat]-dVect[[i]],{i,1,m}];
  uVect = LinearSolve[aMat,bVect]; uMat = pMat;
  Do[uMat = uMat + uVect[[i]] fMat[i], {i,1,m}];
  vPwMat  =  invxMat.(rMat - uMat.yMat);
  vMat    = (vPwMat + Transpose[vPwMat]) / 2;
  Return[{uVect,uMat,vMat}];
];

(** ---> Step 5 **)
getAlphaBar[xMat_,yMat_,uMat_,vMat_] :=
Module[{lambdaInvXU,lambdaInvYV,lambdaMin,alphaBar,nonzero},
  nonzero = 0.01;
  lambdaInvXU = Min[Eigenvalues[Inverse[xMat].uMat]];
  lambdaInvYV = Min[Eigenvalues[Inverse[yMat].vMat]];
  lambdaMin = Min[{lambdaInvXU,lambdaInvYV}];
  If[lambdaMin<-nonzero,alphaBar=-1.0/lambdaMin,alphaBar=100.0];
  Return[alphaBar];
];

getStepLength[n_,xMat_,uMat_,yMat_,vMat_,
                  gammaStar_,xiStar_,theta_,mu0_,mu_] :=
Module[{alpha,alphaBar,alphaHat,uv,a,b,c,d,nonzero},
  nonzero = 1.0 10^-7; alphaHat = 100.0;
  alphaBar = getAlphaBar[xMat,yMat,uMat,vMat];
  If[(theta<nonzero)&&(beta<1.0-nonzero),alphaHat=1.0/(1.0-beta)];
  If[theta >= nonzero,
    uv = innerProduct[n,uMat,vMat];
    a = xiStar uv / n; b = theta mu0 -(1.0 -beta) mu xiStar;
    c = xiStar mu - theta mu0; d = b b - 4.0 a c;
    If[(a<-nonzero)&&(d>0.0),alphaHat=(-b-Sqrt[d])/(2.0 a)];
    If[(Abs[a]<=nonzero)&&(b<-nonzero),alphaHat=-c/b];
    If[(a>nonzero)&&(d>0.0)&&(b<0.0),alphaHat=(-b-Sqrt[d])/(2.0 a)]
  ];
  alpha = Min[{gammaStar alphaBar, alphaHat, 1.0}];
  Return[alpha];
];

(** Printing Sub Routines **)
printParameters[titleAndComment_,pStar_,epsilonStar_,
  betaStar_,xiStar_,gammaStar_,alphaStar_] :=
Module[{},
  Print["* Primitive INterior Point ALgorithm *"];
```

```
  Print["* Masakazu Kojima, Tokyo Institute of Technology *"];
  Print["* Version 1.0, November 29, 1994 *"];
  Print[titleAndComment];
  Print["* Parameters *"]; Print["pStar = ",pStar];
  Print["epsilonStar = ",epsilonStar];
  Print["betaStar = ",betaStar]; Print["gammaStar = ",gammaStar];
  Print["alphaStar = ",alphaStar]; Print["xiStar = ",xiStar];
];

printIteration[pIteration_,theta_,mu_,alpha_,beta_] :=
Module[{},
  Print["* iteration = ",pIteration," *"];
  Print["mu = ",mu,"  theta = ",theta];
  Print["beta = ",beta,"  alpha  = ",alpha];
];

printSolution[m_,n_,cVect_,fMat_,pIteration_,
  xVect_,xMat_,yMat_,mu_,theta_,epsilonStar_] :=
Module[{residualList,pMat,dVect,normOFpMat,normOFdVect,sum,i},
  residualList = getResidual[m,n,cVect,fMat,xVect,xMat,yMat];
  pMat  = residualList[[1]];
  normOFpMat = matrixNorm[n,pMat];
  dVect = residualList[[2]];
  normOFdVect = vectorNorm[m,dVect];
  Print["**** Computational Results *****"];
  Print["Total No. of Iteration = ",pIteration];
  Print["Primal feasibility error = ",normOFpMat];
  Print["Dual   feasibility error = ",normOFdVect];
  Print["mu = ",mu];
  Print["Minimum Eigenvalue of xMat = ", Min[Eigenvalues[xMat]]];
  Print["Minimum Eigenvalue of yMat = ", Min[Eigenvalues[yMat]]];
  Print["Maximum Eigenvalue of xMat.yMat = ",
    Max[Eigenvalues[xMat.yMat]]];
  If[mu < epsilonStar,
    sum = 0.0; Do[sum = sum + cVect[[i]] xVect[[i]],{i,1,m}];
    Print["** Approximate Optimal Solution **"];
    Print["Primal Objective Value = ",sum];
    Print["Dual   Objective Value = ",innerProduct[n,fMat[0],yMat]];
    Print["xVect = ",xVect]; Print["xMat  = ",xMat];
    Print["yMat  = ",yMat],
    Print["** Infeasible? Or Try again with a larger pStar! **"]
  ];
];
```

References

[1] W. F. Alizadeh, "Interior point methods in semidefinite programming with applications to combinatorial optimization," *SIAM J. on Optimization* to appear.

[2] W. F. Alizadeh, J.-P. Haeberly and M. J. Overton, "Primal-dual interior-point methods for semidefinite programming," Computer Science Department, Courant Institute of Mathematical Sciences, New York University, New York, NY, August, 1994.

[3] S. Boyd, L. E. Ghaoui, E. Feron and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory,* (SIAM, Philadelphia, 1994).

[4] M, Grötschel, Lovász and A. Schrijver, "Polynomial algorithms for perfect graphs," *Annals of Discrete Mathematics* 21 (1984) 325-356.

[5] M, Grötschel, Lovász and A. Schrijver, "Relaxations of vertex packing," *Journal of combinatorial Theory, Series B* 40 (1986) 330-343.

[6] M, Grötschel, Lovász and A. Schrijver, *Geometric algorithms and combinatorial optimization* (Springer, New York, 1988).

[7] M. Kojima, S. Shindoh and S. Hara, "Interior-point methods for the monotone linear complementarity problems in symmetric matrices," Research Reports B-282, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152, April 1994.

[8] M. Kojima, S. Kojima and S. Hara, "Linear algebra for semidefinite programming," Research Reports B-290, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152, October 1994.

[9] L. Lovász and A. Schrijver, "Cones of matrices and set functions and 0-1 optimization," *SIAM J. on Optimization* 1 (1991) 166-190.

[10] Y. E. Nesterov and A. S.Nemirovskii, *Interior Point Polynomial Methods in Convex Programming: Theory and Applications* (SIAM, Philadelphia, 1993).

[11] Y. E. Nesterov and M. J. Todd, "Self-scaled cones and interior-point methods in nonlinear programming," Catholic University of Lonvain, Lonvain-la-Neuve, Belgium, 1994.

[12] L. Vandenberghe and S. Boyd, "A primal-dual potential reduction method for problems involving matrix inequalities," *Mathematical Programming, Series B* to appear.