DIMACS Technical Report 2000-01
January 2000

On Formulating Semidefinite Programming Problems as Smooth Convex Nonlinear Optimization Problems

by

Robert J. Vanderbei[1,2]
Dept. of Operations Research and Financial Engineering
Princeton University
Princeton, New Jersey 08903

Hande Yurttan Benson[3]
Dept. of Operations Research and Financial Engineering
Princeton University
Princeton, New Jersey 08903

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, NEC Research Institute and Telcordia Technologies (formerly Bellcore).

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

Consider the diagonal entries d_j, $j = 1, 2, \ldots, n$, of the matrix D in an LDL^T factorization of an $n \times n$ matrix X. As a function of X, each d_j is well-defined on the closed domain of positive semidefinite matrices. We show that these functions are twice continuously differentiable and concave throughout the interior of this domain. Using these facts, we show how to formulate semidefinite programming problems as standard convex optimization problems that can be solved using an interior-point method for nonlinear programming.

1 Introduction

Consider the *canonical nonlinear optimization problem* (NLP):

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & h_i(x) = 0, & i \in \mathcal{E}, \\ & h_i(x) \geq 0, & i \in \mathcal{I}. \end{array} \tag{1}$$

Here, x is a vector in $\mathcal{R}^n$, the sets $\mathcal{E}$ and $\mathcal{I}$ are finite, and each of the functions are defined on $\mathcal{R}^n$ and take values in the extended reals. Henceforth, let $\mathcal{D}$ denote the feasible set for this problem:

$$\mathcal{D} = \{x : h_i(x) = 0 \; \forall i \in \mathcal{E} \text{ and } h_i(x) \geq 0 \; \forall i \in \mathcal{I}\}.$$

The NLP is called *convex* if each of the h_i's appearing in an equality constraint is affine, f is convex, and all of the h_i's in inequality constraints are concave. It is called *smooth* if, in addition, these functions are continuous on $\mathcal{D}$ and twice continuously differentiable in the relative interior of $\mathcal{D}$. The NLP is called *linear* if all of the defining functions are affine.

The following problem is the *canonical semidefinite programming problem* (SDP):

$$\begin{array}{lll} \text{minimize} & f(X) & \\ \text{subject to} & h_i(X) = 0, & i \in \mathcal{E}, \\ & X \succeq 0. & \end{array} \tag{2}$$

Here, X belongs to $\mathcal{S}^{n\times n}$, the space of symmetric $n \times n$-matrices, each of the functions are real-valued affine functions on $\mathcal{S}^{n\times n}$, and the notation $X \succeq 0$ means that X is a positive semidefinite matrix.

The question that motivates the work reported in this paper is: to what extent can one express SDP's as smooth convex NLP's? In the next section, we first show that two simple ideas on how to do this don't quite work and then we focus our attention on the entries d_j of the diagonal matrix D in an LDL^T factorization of X and re-express the semidefiniteness constraint $X \succeq 0$ of SDP with n inequality constraints: $d_j \geq 0$, $j = 1, 2, \ldots, n$.

In Section 3, we show that these diagonals, as functions of X, are twice differentiable and concave throughout the interior of the domain of positive semidefinite matrices. In Section 4, we describe a generic interior-point algorithm for NLP and, in Section 5, we show how to modify it slightly to make it applicable to this NLP formulation of an SDP. In Section 6, we present some preliminary numerical results and finally, in Section 7, we compare the algorithm described in this paper with the usual interior-point algorithms for SDP.

Some of the early ideas on how to express semidefinite matrix constraints in the NLP context date back to the early 1980's with Fletcher's work [8], which was motivated by the *educational testing problem* [7]. In [8], Fletcher gives various characterizations for the normal cone and the set of feasible directions associated with a semidefiniteness constraint. Using these characterizations he develops an SQP method for a certain class of SDP's. More recently, Burer, et.al., give in [6, 5] a rather different approach to solving the problem considered here. In [4], Burer and Monteiro replace X with LL^T and treat the lower triangular matrix L as the new set of variables in the problem. They show that this approach can be applied effectively to solve the SDP relaxation of the max-cut problem. For a rather complete survey of the state of the art as of 1996, the reader is refered to Lewis and Overton [9].

2 Characterizations of Semidefiniteness

To express an SDP as an NLP, we only need to replace the constraint that X be semidefinite with an equivalent statement involving equalities and/or inequalities. There are a few choices on how to do this.

2.1 Definition of Semidefiniteness—Semi-infinite LP

The most obvious characterization is simply the definition of positive semidefiniteness:

$$\xi^T X \xi \geq 0 \qquad \forall \xi \in \mathcal{R}^n. \tag{3}$$

These constraints are inequalities but there is an uncountably infinite number of them. While it is easy to reduce this to a countably infinite number, it is impossible to get a finite number of constraints this way. Nonetheless, one could take a finite number, say $2n^2$, to get an LP relaxation to the SDP. After solving the LP relaxation, it is easy to generate a new ξ whose constraint is violated by the optimal solution to the LP relaxation, add this to the finite collection (perhaps deleting the most nonbinding of the current constraints at the same time) and solve the new LP relaxation. Continuing in this way, one can produce a sequence of LP's whose solutions converge to the optimal solution to the SDP.

2.2 Eigenvalues—Nonsmooth NLP

A second characterization of semidefiniteness is the condition that all of the eigenvalues of X be nonnegative. Of course, since eigenvalues are enumerated in sorted order, it suffices to stipulate just that the smallest be nonnegative. Let $\lambda_j(X)$ denote the j-th smallest eigenvalue of X. The condition then is

$$\lambda_1(X) \geq 0. \tag{4}$$

As a function from symmetric matrices into the reals, λ_1 is concave (and therefore continuous) but it is not continuously differentiable (the other eigenvalues are even worse: they are not even concave functions). A proof that λ_1 is concave can be found in [12]. To show that it is not smooth (and that the other eigenvalues are not even concave), it suffices to consider the $n = 2$ case:

$$X = \begin{bmatrix} x_1 & y \\ y & x_2 \end{bmatrix}.$$

In this case, the eigenvalues can be given explicitly:

$$\begin{aligned} \lambda_1(X) &= \left((x_1 + x_2) - \sqrt{(x_1 - x_2)^2 + 4y^2} \right) /2 \\ \lambda_2(X) &= \left((x_1 + x_2) + \sqrt{(x_1 - x_2)^2 + 4y^2} \right) /2. \end{aligned}$$

It is easy to check by explicitly differentiating that λ_1 is strictly concave, that λ_2 is strictly convex, and that these functions are infinitely differentiable except where the argument of the square root vanishes, which is precisely where $\lambda_1 = \lambda_2$.

To those who might hope that the nonsmoothness of λ_1 is an inconvenience that can be ignored without negative impact, consider the fact that for linear programming the analogous situation arises if one replaces the componentwise nonnegativity of a vector x with the single constraint $\min(x_j) \geq 0$. Suddenly the nonsmoothness seems unignorable. To see this connection with linear programming, consider the case where X is a diagonal $n \times n$ matrix: $X = \mathrm{Diag}(x)$. Here, $x = (x_1, x_2, \ldots, x_n)$ belongs to $\mathcal{R}^n$. The j-th eigenvalue is the j-th smallest component, $x_{(j)}$, of the vector x: $\lambda_j(X) = x_{(j)}$. While the j-th coordinate of the vector x is a smooth linear function of x, the j-th smallest component of x is neither linear nor smooth. It is the sorting of the components that produces this nonsmoothness and nonlinearity.

2.3 Factorization—Smooth Convex NLP

A third characterization of semidefiniteness is based on the fact that each positive semidefinite matrix X can be factored into the product of three matrices: a unit lower triangular matrix L, a diagonal matrix D, and the transpose of L: $X = LDL^T$. It is well-known (see, e.g., [10]) that this factorization exists and D is unique on the domain of positive semidefinite matrices (L is unique only when X is positive definite).

Let d_j denote the j-th diagonal element of D viewed as a function defined on the space of symmetric positive semidefinite matrices. While matrix factorization has a natural extension to the (larger) domain of all symmetric matrices having nonsingular principle submatrices, we nonetheless extend the definition of d_j to all symmetric matrices by putting $d_j(X) = -\infty$ whenever X is not positive semidefinite. With this definition, we shall show that each of the d_j's is concave. They are also each twice continuously differentiable on the set of positive definite matrices.

If the nonlinear programming algorithm that is used to solve the SDP is initialized with a positive definite matrix X and is such that it preserves this property from one iteration to the next, then the constraints

$$d_j(X) \geq 0, \qquad j = 1, 2, \ldots, n, \tag{5}$$

can be used in place of $X \succeq 0$.

We will show in the next section that the d_j's are concave.

3 The Concavity of the d_j's.

In this section we show that each diagonal element of D in an LDL^T factorization of a positive definite matrix X is concave in X.

Let $\mathcal{S}_+^{n\times n}$ denote the set of symmetric positive semidefinite matrices and let $\mathcal{S}_{++}^{n\times n}$ denote the subset of $\mathcal{S}_+^{n\times n}$ consisting of the positive definite matrices. We endow these sets with the topology induced by the Frobenius matrix norm defined by $\|X\|^2 = \sum_{ij} x_{ij}^2$. The following results are well-known.

Theorem 1

1. *The interior of $\mathcal{S}_+^{n\times n}$ is $\mathcal{S}_{++}^{n\times n}$ (see, e.g., [3], p. 20).*
2. *For every X in $\mathcal{S}_+^{n\times n}$ there exists a unit triangular matrix L and a unique diagonal matrix D for which $X = LDL^T$ (see, e.g., [18]).*

Fix $j \in \{1, 2, \ldots, n\}$ and let x, y, Z denote the following blocks of X:

$$X = \begin{array}{cc} & \begin{array}{ccc} & j & \end{array} \\ \begin{array}{c} \\ j \\ \\ \end{array} & \left[\begin{array}{c|c|c} Z & y & * \\ \hline y^T & x & * \\ \hline * & * & * \end{array}\right] \end{array}. \tag{6}$$

Theorem 2 *For $X \in \mathcal{S}_{++}^{n\times n}$, the matrix Z is nonsingular and $d_j(X) = x - y^T Z^{-1} y$.*

Proof 1 *It is easy to check that every principle submatrix of a positive definite matrix is itself positive definite. Therefore Z is positive definite and hence nonsingular. Now, factor X into LDL^T and partition L and D as we did X:*

$$L = \begin{array}{cc} & \begin{array}{ccc} & j & \end{array} \\ \begin{array}{c} \\ j \\ \\ \end{array} & \left[\begin{array}{c|c|c} L_0 & 0 & 0 \\ \hline u^T & 1 & 0 \\ \hline * & * & * \end{array}\right] \end{array}, \qquad D = \begin{array}{cc} & \begin{array}{ccc} & j & \end{array} \\ \begin{array}{c} \\ j \\ \\ \end{array} & \left[\begin{array}{c|c|c} D_0 & 0 & 0 \\ \hline 0 & d & 0 \\ \hline 0 & 0 & * \end{array}\right] \end{array}.$$

From $X = LDL^T$, it is easy to check that

$$\begin{aligned} Z &= L_0 D_0 L_0^T \\ y &= L_0 D_0 u \qquad (7) \\ x &= d + u^T D_0 u. \qquad (8) \end{aligned}$$

From (7), we see that $u = D_0^{-1}L_0^{-1}y$. *Substituting this expression into (8), we get*

$$\begin{aligned} x &= d + y^T L_0^{-T} D_0^{-1} L_0^{-1} y \\ &= d + y^T Z^{-1} y. \end{aligned}$$

Theorem 3 *The function* $d : \mathcal{R} \times \mathcal{R}^n \times \mathcal{S}_{++}^{n\times n} \longrightarrow \mathcal{R}$ *defined by* $d(x, y, Z) = x - y^T Z^{-1} y$ *is concave.*

Proof 2 *It suffices to show that* $f(y, Z) = y^T Z^{-1} y$ *is convex in* (y, Z). *To do so, we look at the first and second derivatives, which are easy to compute using the identity* $\partial Z^{-1}/\partial z_{ij} = -Z^{-1} e_i e_j^T Z^{-1}$:

$$\begin{aligned} \frac{\partial f}{\partial y_i} &= y^T Z^{-1} e_i + e_i^T Z^{-1} y, \\ \frac{\partial^2 f}{\partial y_i \partial y_j} &= e_j^T Z^{-1} e_i + e_i^T Z^{-1} e_j, \\ \frac{\partial f}{\partial z_{ij}} &= -y^T Z^{-1} e_i e_j^T Z^{-1} y, \\ \frac{\partial^2 f}{\partial z_{ij} \partial z_{kl}} &= y^T Z^{-1} e_k e_l^T Z^{-1} e_i e_j^T Z^{-1} y + y^T Z^{-1} e_i e_j^T Z^{-1} e_k e_l^T Z^{-1} y, \\ \frac{\partial^2 f}{\partial y_i \partial z_{kl}} &= -y^T Z^{-1} e_k e_l^T Z^{-1} e_i - e_i^T Z^{-1} e_k e_l^T Z^{-1} y. \end{aligned}$$

Letting H *denote the Hessian of* f *(with respect to each of the* z_{ij}*'s and the* y_i*'s), we compute* $\xi^T H \xi$ *using the above values, where* ξ *is a vector of the form*

$$\xi = \left[\begin{array}{cccccccccccccc} a_{11} & a_{12} & \cdots & a_{1n} & \cdots & a_{n1} & a_{n2} & \cdots & a_{nn} & b_1 & b_2 & \cdots & b_n \end{array} \right]^T.$$

Letting $A = [a_{ij}]$ *and* $b = [b_i]$, *we get that*

$$\begin{aligned} \xi^T H \xi &= \sum_{i,j,k,l} a_{ij} \left(y^T Z^{-1} e_k e_l^T Z^{-1} e_i e_j Z^{-1} y + y^T Z^{-1} e_i e_j^T Z^{-1} e_k e_l Z^{-1} y \right) a_{kl} \\ &\quad + \sum_{i,k,l} b_i \left(-y^T Z^{-1} e_k e_l^T Z^{-1} e_i - e_i^T Z^{-1} e_k e_l^T Z^{-1} y \right) a_{kl} \\ &\quad + \sum_{i,j,l} a_{ij} \left(-y^T Z^{-1} e_i e_j^T Z^{-1} e_l - e_l^T Z^{-1} e_i e_j^T Z^{-1} y \right) b_l \\ &\quad + \sum_{i,j} b_i \left(e_j^T Z^{-1} e_i + e_i^T Z^{-1} e_j \right) b_j \\ &= 2 \left(y^T Z^{-1} A Z^{-1} A Z^{-1} y - y^T Z^{-1} A Z^{-1} b - b^T Z^{-1} A Z^{-1} y + b^T Z^{-1} b \right). \end{aligned}$$

Since Z *is symmetric and positive definite, we can assume that* A *is symmetric too and we get*

$$\begin{aligned} \xi^T H \xi &= 2 \left(y^T Z^{-1} A Z^{-1/2} - b^T Z^{-1/2} \right) \left(Z^{-1/2} A Z^{-1} y - Z^{-1/2} b \right) \\ &= 2 \left\| Z^{-1/2} (A Z^{-1} y - b) \right\|^2 \\ &\geq 0. \end{aligned}$$

Thus, H is positive semidefinite, and f is convex.

Remark. The expressions for the derivatives with respect to the diagonal elements of X were also given by Fletcher in [8].

Since $\det(D) = \det(LDL^T) = \det(X)$, it follows that $-\sum_{j=1}^n \log(d_j(X))$ is the usual self-concordant barrier function for SDP (see, e.g., [11]).

4 An interior-point method for convex optimization.

In this section, we describe an interior-point algorithm that can be easily modified to handle SDPs in which the positive semidefiniteness constraint is expressed using (5). We describe the algorithm for problems in which all constraints are in inequality form:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & h_i(x) \geq 0, \qquad i = 1, \ldots, m. \end{array} \tag{9}$$

We restrict our attention to this case purely for notational brevity—the implementation which we discuss in Section 7 allows equality constraints and even constraints with ranges. A complete description of the algorithm in the most general case can be found in [15]. On the feasible set, we assume that the objective function f is convex and that each inequality constraint function is concave.

We derive an interior-point method for this problem as follows. First we subtract surplus variables from all of the constraints to rewrite them as

$$h_i(x) - w_i = 0.$$

Of course, each variable w_i is assumed to be nonnegative.

Next, we write each of the x_j's as the difference of two nonnegative variables $x_j = g_j - t_j$. The reason for doing this is to make the dual normal equations (which will be defined later) into a positive definite system when it would otherwise be only positive semidefinite.

In this way, with extra variables and extra constraints, we reformulate the problem so that all constraints are equalities and all of the newly added variables are nonnegative:

$$\begin{array}{llll} \text{minimize} & f(x) & & \\ \text{subject to} & h_i(x) - w_i = 0, & & i = 1, \ldots, m, \\ & x_j - g_j + t_j = 0, & & i = 1, \ldots, n, \\ & w_i \geq 0, & & i = 1, \ldots, m, \\ & g_j \geq 0, & & j = 1, \ldots, n, \\ & t_j \geq 0, & & j = 1, \ldots, n. \end{array}$$

We next replace the nonnegativity constraints with logarithmic barrier terms in the objective

$$\begin{array}{ll} \text{minimize} & f(x) - \mu \sum_i \log(w_i) - \mu \sum_j \log(g_j) - \mu \sum_j \log(t_j) \\ \text{subject to} & h_i(x) - w_i = 0, \qquad i = 1, \dots, m \\ & x_j - g_j + t_j = 0, \qquad i = 1, \dots, n. \end{array}$$

The Lagrangian for this problem is

$$\begin{aligned} L(x, w, g, t, y, z) \;&=\; f(x) - \mu \sum_i \log(w_i) - \mu \sum_j \log(g_j) - \mu \sum_j \log(t_j) \\ &\quad - \sum_i y_i \left(h_i(x) - w_i\right) - \sum_j z_j \left(x_j - g_j + t_j\right). \end{aligned}$$

Letting $h(x)$ denote the vector whose components are $h_i(x)$, the first-order conditions for a minimum are

$$\begin{aligned} \nabla_x L &= \nabla f(x) - \nabla h(x)^T y - z = 0, \\ \nabla_w L &= -\mu W^{-1} e + y = 0, \\ \nabla_g L &= -\mu G^{-1} e + z = 0, \\ \nabla_t L &= -\mu T^{-1} e - z = 0, \\ \nabla_y L &= h(x) - w = 0, \\ \nabla_z L &= x - g + t = 0, \end{aligned} \tag{10}$$

where W, G, T are diagonal matrices with elements w_i, g_i, t_i, respectively, e is the vector of all ones, and $\nabla h(x)$ is the $m \times n$ Jacobian matrix of the vector $h(x)$. Next, inspired by ideas of primal-dual complementarity, we introduce new variables s defined by

$$-\mu T^{-1} e + s = 0$$

and replace the $\mu T^{-1} e$ expression in (10) with s. We then multiply through by W, G, and T to get the following primal-dual system

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y - z &= 0, \\ -\mu e + WYe &= 0, \\ -\mu e + GZe &= 0, \\ -\mu e + TSe &= 0, \\ -s - z &= 0, \\ h(x) - w &= 0, \\ x - g + t &= 0. \end{aligned} \tag{11}$$

To this nonlinear system, we apply Newton's method to get step directions for each of the variables. In order to simplify notation in the Newton system and at the same time highlight connections with linear and quadratic programming, we introduce the following definitions:

$$H(x, y) = \nabla^2 f(x) - \sum_{i=1}^{m} y_i \nabla^2 h_i(x)$$

and

$$A(x) = \nabla h(x) = [\partial h_i(x)/\partial x_j] \, .$$

By appropriately arranging the variables and equations, the Newton system can then be written in the following symmetric form:

$$\left[\begin{array}{cccc|ccc}
 & & & & & I & I \\
 & -G^{-1}Z & & & & -I & \\
 & & -W^{-1}Y & & -I & & \\
 & & & -H(x,y) & A^T(x) & I & \\
\hline
 & & -I & A(x) & & & \\
I & -I & & I & & & \\
I & & & & & & S^{-1}T
\end{array}\right]
\left[\begin{array}{c} \Delta t \\ \Delta g \\ \Delta w \\ \Delta x \\ \hline \Delta y \\ \Delta z \\ \Delta s \end{array}\right]
=
\left[\begin{array}{r} \tau \\ -\gamma_z \\ -\gamma_y \\ \sigma \\ \hline \rho \\ \nu \\ S^{-1}T\gamma_s \end{array}\right],$$

where

$$\begin{aligned}
\rho &= -h(x) + w \\
\sigma &= \nabla f(x) - A^T y - z \\
\tau &= -s - z \\
\nu &= -x + g - t
\end{aligned}$$

record primal and dual constraint infeasibilities and

$$\begin{aligned}
\gamma_z &= \mu G^{-1} e - z \\
\gamma_y &= \mu W^{-1} e - y \\
\gamma_s &= \mu T^{-1} e - s
\end{aligned}$$

capture the deviation from centrality. Using the pivot elements $-G^{-1}Z$, $-W^{-1}Y$, and $S^{-1}T$ to solve for Δg, Δw, and Δs, respectively, we get

$$\begin{aligned}
\Delta g &= GZ^{-1}(\gamma_z - \Delta z) \\
\Delta w &= WY^{-1}(\gamma_y - \Delta y) \\
\Delta s &= \gamma_s - ST^{-1}\Delta t,
\end{aligned} \tag{12}$$

which we then eliminate from the remaining equations to obtain

$$\left[\begin{array}{cc|cc}
-ST^{-1} & & & I \\
 & -H(x,y) & A^T(x) & I \\
\hline
 & A(x) & WY^{-1} & \\
I & I & & GZ^{-1}
\end{array}\right]
\left[\begin{array}{c} \Delta t \\ \Delta x \\ \hline \Delta y \\ \Delta z \end{array}\right]
=
\left[\begin{array}{c} \tau - \gamma_s =: \hat{\tau} \\ \sigma \\ \hline \rho + WY^{-1}\gamma_y \\ \nu + GZ^{-1}\gamma_z =: \hat{\nu} \end{array}\right].$$

Next, we use the pivot element ST^{-1} to solve for Δt to get

$$\Delta t = S^{-1}T(\Delta z - \hat{\tau}) \tag{13}$$

Eliminating these variables from the remaining equations, we get

$$\left[\begin{array}{c|cc} -H(x,y) & A^T(x) & I \\ \hline A(x) & WY^{-1} & \\ I & & GZ^{-1}+S^{-1}T \end{array}\right] \left[\begin{array}{c} \Delta x \\ \hline \Delta y \\ \Delta z \end{array}\right] = \left[\begin{array}{c} \sigma \\ \hline \rho + WY^{-1}\gamma_y \\ \hat{\nu} + S^{-1}T\hat{\tau} \end{array}\right]. \tag{14}$$

Now, we use the pivot element $GZ^{-1} + S^{-1}T$ to solve for Δz:

$$\Delta z = D\left(\hat{\nu} S^{-1} T\hat{\tau} - \Delta x\right) \tag{15}$$

where

$$D = \left(S^{-1}T + GZ^{-1}\right)^{-1}. \tag{16}$$

Eliminating Δz from (14), we arrive at last at the *reduced KKT system*:

$$\left[\begin{array}{c|c} -(H+D) & A^T \\ \hline A & WY^{-1} \end{array}\right] \left[\begin{array}{c} \Delta x \\ \hline \Delta y \end{array}\right] = \left[\begin{array}{c} \sigma - D(\hat{\nu} + S^{-1}T\hat{\tau}) \\ \hline \rho + WY^{-1}\gamma_y \end{array}\right]. \tag{17}$$

Note that we no longer mention explicitly the dependence of H and A on x and y. From the reduced KKT system, we use pivot element WY^{-1} to solve for Δy:

$$\Delta y = W^{-1}Y\rho + \gamma_y - W^{-1}YA\Delta x. \tag{18}$$

Eliminating Δy from the reduced KKT system, we get the *dual normal equations*:

$$-N\Delta x = \sigma - D(\hat{\nu} + S^{-1}T\hat{\tau}) - A^TWY^{-1}\rho - A^T\gamma_y, \tag{19}$$

where

$$N = D + H + A^TW^{-1}YA.$$

Working backwards through (19), (18), (15), (13), and (12), we solve for Δx, Δy, Δz, Δt, Δg, Δw, and Δs.

The algorithm can now be summarized as follows. Start with values for x, y, z, t, g, w, and s having the property that y, z, t, g, w, and s are componentwise strictly positive. Compute using some heuristic a reasonable value for μ. Compute step directions as described above. Compute the largest value of $\alpha \in [0,1]$ such that the nonnegative variables at the new point $x + \alpha\Delta x$, $y + \alpha\Delta y$, $z + \alpha\Delta z$, $t + \alpha\Delta t$, $g + \alpha\Delta g$, $w + \alpha\Delta w$, $s + \alpha\Delta s$ will be nonnegative and then shorten it to preserve strict positivity and shorten it even more, if necessary, to get a decrease in a certain merit function (see [17] for details on a merit function). Make this new point the current point and repeat the process until all infeasibilities are vanishingly small and primal-dual complementarity is essentially achieved.

The algorithm just described is precisely how the first author's software, LOQO, solves problem (9). We repeat that this software package can solve problems with more general formulations, such as with the equality constraints in (1), but for notational simplicity we focus on the simpler form of (9).

For later reference it is useful to have explicit formulas for Δx and Δw. To this end, we substitute the definitions of σ, $\hat{\nu}$, etc., into (19) and do some algebraic simplification to get

$$\begin{aligned}\Delta x &= -N^{-1}\nabla f(x) \\ &\quad +N^{-1}A^TW^{-1}Y\rho + N^{-1}D\nu \\ &\quad +\mu N^{-1}\left(DZ^{-1}e - DS^{-1}e + A^TW^{-1}e\right).\end{aligned}$$

In this expression for Δx, the term on the first line is the component of the step direction that aims toward optimality, the terms on the second line aim toward feasibility, and the terms on the third line provide centering. From the relation $\Delta w = -\rho + A\Delta x$, we see that

$$\begin{aligned}\Delta w &= -AN^{-1}\nabla f(x) \\ &\quad -\left(I - AN^{-1}A^TW^{-1}Y\right)\rho + AN^{-1}D\nu \\ &\quad +\mu AN^{-1}\left(DZ^{-1}e - DS^{-1}e + A^TW^{-1}e\right).\end{aligned}$$

The interior-point algorithm just presented has the property that it maintains strict positivity of the artificial variables. But it does not necessarily maintain positivity of the constraint functions. We need this property in order to apply this algorithm to SDPs formulated as NLPs using constraints of the form (5).

In the next section, we address those algorithmic modifications that are necessary in order to maintain constraint function positivity.

5 Preserving strict positivity of the constraint functions.

Sometimes it is easy to initialize x so that $h_i(x) > 0$. Such is the case with the constraints defining positive definiteness. The question then arises, does the algorithm given in the previous section preserve this property from one iteration to the next or, if not, is it possible to modify the algorithm to preserve it? The answer is that it doesn't preserve the property automatically but that shortened steps can be used to accomplish it.

The function h_i is related to the infeasibility ρ_i as follows:

$$h_i(x) = w_i - \rho_i$$

(of course, ρ_i depends on x and w even though we don't show it). To conclude that $h_i(x) > 0$, it would suffice to show that $\rho_i \leq 0$. In linear programming, the infeasibility measure changes by a simple scaling from one iteration to the next and hence if ρ_i is nonnegative initially then it will remain nonnegative throughout the iterations.

Perhaps a similar relation will hold here. An algebraic rearrangement of the Newton equation associated with the i-th constraint yields

$$\nabla h_i(x)^T \Delta x = \Delta w_i + \rho_i.$$

Using the concavity of h_i, we bound the infeasibility $\tilde{\rho}_i$ at the next iteration:

$$\begin{aligned} \tilde{\rho}_i &= w_i + \alpha \Delta w_i - h_i(x + \alpha \Delta x) \\ &\geq w_i + \alpha \Delta w_i - h_i(x) - \alpha \nabla h_i(x)^T \Delta x \\ &= w_i - h_i(x) - \alpha \left(\nabla h_i(x)^T \Delta x - \Delta w_i \right) \\ &= (1 - \alpha) \rho_i. \end{aligned}$$

We end up with the same sort of relation as one finds in linear programming except that the equality from LP has been replaced with an inequality. With this inequality, we see that if at some iteration ρ_i is positive, then at every subsequent iteration it remains positive. This is precisely the opposite of what we wanted.

In fact, computational experiments with SDP verify that ρ_i does indeed go positive. It even surpasses w_i and therefore h_i goes negative.

To prevent h_i from going negative, we can shorten the step length α appropriately. With shortening we can guarantee that $h_i(x)$ remains positive at every iteration. Given that h_i is positive at the beginning of each iteration, we can also reset w_i equal to h_i so that ρ_i is zero. These changes to the algorithm will give us the desired positivity of h_i but the resulting algorithm could have the undesirable "jamming" behavior in which the iterates jam into a "wall" and fail to make further progress toward optimality. A necessary anti-jamming condition is that each component of the vector field of step directions is positive in a neighborhood of the set where the corresponding component of the solution vector vanishes.

To be precise, we consider a point $(\bar{x}, \bar{y}, \bar{z}, \bar{t}, \bar{g}, \bar{w}, \bar{s})$ satisfying the following conditions:

1. Nonnegativity: $\bar{y} \geq 0$, $\bar{z} \geq 0$, $\bar{t} \geq 0$, $\bar{g} \geq 0$, $\bar{w} \geq 0$, and $\bar{s} \geq 0$.
2. Complementary variable positivity: $\bar{w} + \bar{y} > 0$, $\bar{g} + \bar{z} > 0$, and $\bar{t} + \bar{s} > 0$.
3. Free variable positivity: $\bar{g} + \bar{t} > 0$.

Of course, we are interested in a point where some of the $\bar{w}_i$'s vanish. Let $\mathcal{U}$ denote the set of constraint indices for which $\bar{w}_i$ vanishes and let $\mathcal{B}$ denote those for which it doesn't. Write A (and other matrices and vectors) in block form according to this partition

$$A = \begin{bmatrix} B \\ U \end{bmatrix}$$

and let

$$K = N - U^T W_{\mathcal{U}}^{-1} Y_{\mathcal{U}} U = D + H + B^T W_{\mathcal{B}}^{-1} Y_{\mathcal{B}} B.$$

Matrices A and K as well as other quantities are functions of the current point. We use the same letter with a bar over it to denote the value of these objects at the point $(\bar{x}, \ldots, \bar{s})$.

Theorem 4 *If the point $(\bar{x}, \bar{y}, \bar{z}, \bar{t}, \bar{g}, \bar{w}, \bar{s})$ satisfies conditions (1)–(3), $\bar{U}$ has full row rank, and $\bar{K}$ is positive definite, then Δw has a continuous extension to this point and $\Delta w_{\mathcal{U}} = \mu Y_{\mathcal{U}}^{-1} e_{\mathcal{U}} > 0$ there.*

Proof 3 *From the formula for Δw given in the previous section, we see that*

$$\begin{aligned} \Delta w_{\mathcal{U}} + \rho_{\mathcal{U}} &= UN^{-1}\left(-\nabla f(x) + D\nu + \mu D Z^{-1} e - \mu D S^{-1} e\right) \\ &\quad + UN^{-1} A^T W^{-1} (Y\rho + \mu e). \end{aligned} \tag{20}$$

To prove the theorem, we must analyze the limiting behavior of UN^{-1} and $UN^{-1}A^TW^{-1}$.

Applying the Sherman–Morrison–Woodbury formula, we get

$$\begin{aligned} N^{-1} &= (U^T W_{\mathcal{U}}^{-1} Y_{\mathcal{U}} U + K)^{-1} \\ &= K^{-1} - K^{-1} U^T \left(UK^{-1}U^T + W_{\mathcal{U}} Y_{\mathcal{U}}^{-1}\right)^{-1} UK^{-1}. \end{aligned}$$

Hence,

$$\begin{aligned} UN^{-1} &= UK^{-1} - UK^{-1}U^T \left(UK^{-1}U^T + W_{\mathcal{U}} Y_{\mathcal{U}}^{-1}\right)^{-1} UK^{-1} \\ &= W_{\mathcal{U}} Y_{\mathcal{U}}^{-1} \left(UK^{-1}U^T + W_{\mathcal{U}} Y_{\mathcal{U}}^{-1}\right)^{-1} UK^{-1}. \end{aligned} \tag{21}$$

From the definition of $\mathcal{U}$, we have that $\bar{W}_{\mathcal{U}} = 0$. In addition assumption (2) implies that $\bar{Y}_{\mathcal{U}} > 0$, which then implies that $Y_{\mathcal{U}}^{-1}$ remains bounded in a neighborhood of $(\bar{x}, \ldots, \bar{s})$. The assumption that $\bar{U}$ has full row rank and that $\bar{K}$ is nonsingular implies that the following limit exists:

$$\lim \left(UK^{-1}U^T + W_{\mathcal{U}} Y_{\mathcal{U}}^{-1}\right)^{-1} UK^{-1} = \left(\bar{U}\bar{K}^{-1}\bar{U}^T\right)^{-1} \bar{U}\bar{K}^{-1}.$$

Here and throughout this proof, all limits are understood to be taken as $(x, \ldots, s)$ approaches $(\bar{x}, \ldots, \bar{s})$. From the previous limit, we see that

$$\lim UN^{-1} = 0.$$

It is easy to check that assumptions (1)–(3) imply that the terms multiplied by UN^{-1} on the first line of (20) remain bounded in the limit and therefore

$$\lim UN^{-1}\left(-\nabla f(x) + D\nu + \mu D Z^{-1} e - \mu D S^{-1} e\right) = 0. \tag{22}$$

Now, consider $UN^{-1}A^TW^{-1}$. Writing A and W in block form and using (21), we get

$$UN^{-1}A^TW^{-1} = W_{\mathcal{U}} Y_{\mathcal{U}}^{-1} \left(UK^{-1}U^T + W_{\mathcal{U}} Y_{\mathcal{U}}^{-1}\right)^{-1} UK^{-1} \left[\begin{array}{cc} B^T W_{\mathcal{B}}^{-1} & U^T W_{\mathcal{U}}^{-1} \end{array}\right].$$

The analysis of UN^{-1} in the previous paragraph applies to the first block of this block matrix and shows that it vanishes in the limit. The analysis of the second block is more delicate because of the $W_{\mathcal{U}}^{-1}$ factor:

$$\begin{aligned} W_{\mathcal{U}} Y_{\mathcal{U}}^{-1} \left(UK^{-1}U^T + W_{\mathcal{U}} Y_{\mathcal{U}}^{-1} \right)^{-1} UK^{-1}U^T W_{\mathcal{U}}^{-1} & \\ &= W_{\mathcal{U}} Y_{\mathcal{U}}^{-1} \left(I - \left(UK^{-1}U^T + W_{\mathcal{U}} Y_{\mathcal{U}}^{-1} \right)^{-1} W_{\mathcal{U}} Y_{\mathcal{U}}^{-1} \right) W_{\mathcal{U}}^{-1} \\ &= \left(I - W_{\mathcal{U}} Y_{\mathcal{U}}^{-1} \left(UK^{-1}U^T + W_{\mathcal{U}} Y_{\mathcal{U}}^{-1} \right)^{-1} \right) Y_{\mathcal{U}}^{-1} \\ &= UK^{-1}U^T \left(UK^{-1}U^T + W_{\mathcal{U}} Y_{\mathcal{U}}^{-1} \right)^{-1} Y_{\mathcal{U}}^{-1} \end{aligned}$$

From this last expression, we see that the limiting value for the second block is just $Y_{\mathcal{U}}^{-1}$. Putting the two blocks together, we get that

$$\lim UN^{-1}A^T W^{-1} = \left[\begin{array}{cc} 0 & \bar{Y}_{\mathcal{U}}^{-1} \end{array} \right]$$

and hence that

$$\lim UN^{-1}A^T W^{-1} \left(Y\rho + \mu e \right) = \bar{\rho}_{\mathcal{U}} + \mu \bar{Y}_{\mathcal{U}}^{-1} e_{\mathcal{U}}. \tag{23}$$

Combining (22) and (23), we see that

$$\lim \Delta w_{\mathcal{U}} = \mu \bar{Y}_{\mathcal{U}}^{-1} e_{\mathcal{U}}.$$

It follows from Theorem 4 that the interior-point algorithm will not jam provided that the sequence stays away from boundary points where complimentary pairs of variables both vanish. For linear and quadratic programming, the sequence of iterates does indeed obey this strict complementarity property (see, e.g., [1]) so one would expect it to hold for NLP too, although we don't prove it here.

6 Preliminary computational results.

The interior-point algorithm described in Section 4 is implemented in the first author's code LOQO. To this code, we added a boolean parameter `sdp` which allows one to assert that a problem is an SDP. When this parameter is set, step lengths are shortened so that each nonlinear constraint that starts out positive is guaranteed to remain positive and the corresponding slack variable w_i is adjusted at the beginning of each iteration to guarantee that $\rho_i = 0$. These are the only changes to LOQO.

The simplest way to convey problems to LOQO is via the AMPL modeling language. AMPL doesn't have a built-in nonlinear function for the d_j's but it does provide a mechanism whereby the user can define an arbitrary nonlinear linear function in a dynamically linked library and AMPL will then recognize this function. We implemented such a function for the d_j's called `kth_diag`. Using this function, an AMPL model for the following SDP is easy to write—see Figure 1:

$$\begin{array}{lll} \text{minimize} & \sum_{ij} C_{ij} X_{ij} & \\ \text{subject to} & \sum_{ij} A_{ij}^{(k)} X_{ij} = b_k, & k = 1, 2, \ldots, m \\ & X \succeq 0. & \end{array} \tag{24}$$

We've made four data sets: a small problem suggested by Masakazu Muramatsu, a max cut problem, and two max min eigenvalue problems. The data for these four problems can be downloaded from our web site [14]. Without setting parameter `sdp` to true, the nonlinear functions in `kth_diag` go negative and the algorithm fails on all four problems. However, with the parameter set to true, LOQO correctly each of these four problems.

6.1 Exploiting Sparsity

The dual problem for (24) is

$$\begin{array}{lll} \text{maximize} & \sum_{k} b_k y_k & \\ \text{subject to} & \sum_{k} A_{ij}^{(k)} y_k - Z_{ij} = C_{ij}, & i, j = 1, 2, \ldots, n \\ & Z \succeq 0. & \end{array} \tag{25}$$

Suppose that C and each of the $A^{(k)}$ are sparse matrices, It is easy to see from the equality constraints that the sparsity pattern for Z is the union of the sparsity patterns for C and each of the $A^{(k)}$. Therefore, Z is likely to be a sparse matrix too. Since the sparsity patterns are known from the beginning, we can symmetrically permute the rows/columns of these matrices so that the matrix L in an LDL^T factorization of Z has an approximately minimal fill-in. The formulas for $d_j(Z)$ and its derivatives can then exploit this sparsity. Hence, if we replace the constraint $Z \succeq 0$ with the inequalities $d_j(Z) \geq 0$, $j = 1, 2, \ldots, n$ in (25) we get a formulation of the SDP that can exploit sparsity rather nicely. We will report on computational results for this dual formulation in a future paper.

This idea of looking at the dual to exploit sparsity in Z was first suggested and studied by Benson, et.al., in [2].

7 Comparison with Other Interior-Point Methods for SDP.

Consider the derivation of the interior-point method given in Section 4 as it applies to SDPs. Ignoring the special treatment of free variables, the derivation starts by

```
function kth_diag;

param eps, default 0;

param n;
param m;

set N := {1..n};
set M := {1..m};

param A{i in N,j in N,M: j>=i}, default 0;
param C{i in N,j in N:   j>=i}, default 0;
param b{M}, default 0;

var X{i in N,j in N: j>=i};

minimize cost: sum {i in N, j in N: j>i} 2*C[i,j]*X[i,j]
                          + sum {j in N}   C[j,j]*X[j,j];

subject to equalities{k in M}:
    sum {i in N, j in N: j>i} 2*A[i,j,k]*X[i,j]
               + sum {j in N}   A[j,j,k]*X[j,j]
    = b[k];

subject to pos_kth_diag{k in N}:
    kth_diag({i in N, j in N: j>=i} X[i,j]) >= eps;

data sdp.dat;

let {i in N, j in N: j>i} X[i,j] := 0;
let {i in N} X[i,i] := 1;

solve;

display X;
display equalities;
```

Figure 1: An AMPL model for SDP.

replacing $X \succeq 0$ with n inequality constraints to which slack variables are added to get constraints of the form $d_j(X) - w_j = 0$, $w_j \geq 0$. Logarithmic barrier terms for the slack variables are then added to the objective function and first-order optimality conditions are written for the barrier problem. An alternative approach would be to place barriers directly on the nonlinear functions $d_j(X)$ thereby avoiding altogether introducing slack variables. If we follow this second approach, the resulting barrier problem for SDP is:

$$\begin{array}{ll} \text{minimize} & f(X) - \mu \sum_j \log d_j(X) \\ \text{subject to} & h_i(X) = 0, \qquad i \in \mathcal{E}. \end{array}$$

But, as we remarked at the end of Section 3, $\sum_j \log d_j(X) = \log\det X$ and so this barrier problem matches precisely the one used in the "usual" interior-point methods for SDP (see [13] for a survey of these methods). Hence, whether or not one adds slack variables before introducing the barrier problem is the first and principle difference distinguishing the method presented in this paper from other interior-point methods.

How does the addition of slack variables affect the resulting algorithm? To answer this question, let's look at NLPs with only inequality constraints:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & h_i(x) \geq 0, \qquad i \in \mathcal{I}. \end{array}$$

The log barrier problem associated with this NLP is:

$$\text{minimize } f(x) - \mu \sum_i \log h_i(x)$$

The first order optimality conditions then are:

$$\nabla f(x) - \mu \sum_i \frac{\nabla h_i(x)}{h_i(x)} = 0.$$

Applying Newton's method to the first-order optimality conditions, we get the following system for Δx:

$$\left(\nabla^2 f(x) - \mu \sum_i \frac{h_i(x)\nabla^2 h_i(x) - \nabla h_i(x) \nabla h_i^T(x)}{h_i(x)^2} \right) \Delta x = -\nabla f(x) + \mu \sum_i \frac{\nabla h_i(x)}{h_i(x)}. \tag{26}$$

Now, consider what happens if we add slack variables before writing the barrier problem. In this case, the barrier problem is

$$\begin{array}{ll} \text{minimize} & f(x) - \mu \sum_i \log w_i \\ \text{subject to} & h_i(x) - w_i = 0, \qquad i \in \mathcal{I}. \end{array}$$

Using y_i's to denote the Lagrange multipliers, the first-order optimality conditions are:

$$\begin{aligned} \nabla f(x) - \sum_i y_i \nabla h_i(x) &= 0 \\ -\mu W^{-1} e + y &= 0 \\ -h(x) + w &= 0. \end{aligned}$$

Applying Newton's method to these equations, we get the following KKT system:

$$\begin{aligned} \left(\nabla^2 f(x) - \sum_i y_i \nabla^2 h_i(x)\right) \Delta x - \sum_i \nabla h_i(x) \Delta y_i &= -\nabla f(x) + \sum_i y_i \nabla h_i(x) && (27) \\ \mu W^{-2} \Delta w + \Delta y &= \mu W^{-1} - y && (28) \\ -\nabla h(x) \Delta x + \Delta w &= h(x) - w. && (29) \end{aligned}$$

Theorem 5 *If $w = h(x)$ and $y_i w_i = \mu$ for $i \in \mathcal{I}$, then the step direction Δx defined by (27)–(29) coincides with the step direction defined by (26).*

Proof 4 *The assumptions imply that the right-hand sides in (28) and (29) vanish. Use (28) to solve for Dy in terms of Δw and then use (29) to express Δw in terms of Δx. Substitute the resulting formula for Δy in terms of Δx into (27) and the system clearly matches (26).*

Although neither assumption would be expected to hold in general, we have modified the interior-point method when we apply it to SDP in such as way that the first assumption, $w = h(x)$, does indeed hold. But there is no reason for the second one to be satisfied.

What's more, we've left out from our derivation of system (27)–(29) the primal-dual symmetrization step in which the second set of optimality conditions $-\mu W^{-1} w + y = 0$ is replaced with $-\mu e + WYe = 0$. This modification has a significant impact on the step directions producing ones with so-called primal-dual scaling instead of purely primal scaling (see, e.g., [16]).

Of course, for SDP the 'usual" interior-point methods also face a similar issue of using $\mu X^{-1} = Z$ or some primal-dual symmetrization such as $XZ = \mu I$. For SDP, this latter set of equations, while treating the primal and the dual in a symmetric fashion, has the unfortunate property that it destroys matrix symmetry (the product of two symmetric matrices is not symmetric). It is the ambiguity about how to do resolve this issue that has led to the large number of variants of interior-point algorithms for SDP.

Acknowledgements. The authors would like to thank Gabor Pataki, Franz Rendl, and Henry Wolkowicz for discussions that stimulated the work presented here. They also wish to thank Margaret Wright, Michael Overton, and Renato Monteiro for their comments on an earlier draft of this paper.

References

[1] I. Adler and R.D.C. Monteiro. Limiting behavior of the affine scaling continuous trajectories for linear programming. *Mathematical Programming*, 50:29–51, 1991.

[2] S. Benson, Y. Ye, and X. Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. Technical report, Working Paper, Department of Management Sciences, The University of Iowa, 1997. To appear in SIAM J. Optimization.

[3] A. Berman and R.J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences.* Academic Press, 1979.

[4] S. Burer and R.D.C. Monteiro. An efficient algorithm for solving the max-cut (SDP) relaxation. Technical report, School of ISyE, Georgie Tech, Atlanta, GA 30332, USA, 1998. Submitted to *Optimization Methods and Software.*

[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. Technical report, TR99-17, Dept. of Computational and Applied Mathematics, Rice University, Houston TX, 1999.

[6] S. Burer, R.D.C. Monteiro, and Y. Zhang. Solving Semidefinite Programs via Nonlinear Programming Part I: Transformations and Derivatives. Technical report, TR99-17, Dept. of Computational and Applied Mathematics, Rice University, Houston TX, 1999.

[7] R. Fletcher. A nonlinear programming problem in statistics (educational testing). *SIAM J. Sci. Stat. Comput.*, 2:257–267, 1981.

[8] R. Fletcher. Semi-definite matrix constaints in optimization. *SIAM J. Control and Optimization*, 23(4):493–513, 1985.

[9] A.S. Lewis and M.L. Overton. Eigenvalue optimization. *Acta Numerica*, 5:149–190, 1996.

[10] R.S. Martin and J.H. Wilkinson. Symmetric decomposition of positive definite band matrices. *Numer. Math.*, 7:355–361, 1965.

[11] Y.E. Nesterov and A.S. Nemirovsky. *Self–concordant functions and polynomial–time methods in convex programming.* Central Economic and Mathematical Institute, USSR Academy of Science, Moscow, USSR, 1989.

[12] M.L. Overton and R.S. Womersley. Second derivatives for optimizing eigenvalues of symmetric matrices. Technical Report 627, Computer Science Department, NYU, New York, 1993.

[13] M.J. Todd. A study of search directions in primal-dual interior-point methods for semidefinite programming. Technical report, Technical Report No. 1205, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853-3801, 1999.

[14] R.J. Vanderbei. AMPL models. http://www.sor.princeton.edu/~rvdb/ampl/nlmodels.

[15] R.J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 12:451–484, 1999.

[16] R.J. Vanderbei, A. Duarte, and B. Yang. An algorithmic and numerical comparison of several interior-point methods. Technical Report SOR 94-05, Princeton University, 1994.

[17] R.J. Vanderbei and D.F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

[18] J.H. Wilkinson and C. Reinsch. *Handbook for Automatic Computation*, volume II: Linear Algebra. Springer-Verlag, Berlin-Heidelberg-New York, 1971.