COMPARISON BETWEEN AN INFEASIBLE INTERIOR POINT ALGORITHM AND A HOMOGENEOUS SELF DUAL ALGORITHM FOR SEMIDEFINITE PROGRAMMING

by

Qian Xia

Submitted in Partial Fulfillment

of the Requirements for the Degree of

Masters in Mathematics

New Mexico Institute of Mining and Technology

Socorro, New Mexico

Jan. 18, 2006

ABSTRACT

Semidefinite Programming (SDP) involves the optimization of a linear cost function subject to linear constraints over symmetric positive semidefinite matrix variables. One important issue in the design of interior point codes for SDP is the use of an infeasible interior point method versus the use of the homogeneous self dual reformulation. Infeasible interior point methods begin with a solution that may be infeasible with respect to the linear equalities but is feasible with respect to the positive semidefiniteness constraints. Some types of semi-definite programming problems are hard to solve using an infeasible interior point method. For example, if the problem has an unbounded optimal region, most infeasible interior point methods have difficulty. The homogeneous self dual (HSD) reformulation produces a problem with a bounded optimal region, with a known strictly feasible starting point.

In this project, we implement both an infeasible interior point (IIP) and a homogeneous self dual (HSD) algorithm for semi-definite programming. We compare the algorithms in terms of robustness, iteration counts and CPU time.

TABLE OF CONTENTS

LIST OF TABLES **iii**

1. LINEAR and SEMIDEFINITE PROGRAMMING **1**

1.1 Linear Programming 1

1.2 Semidefinite programming 3

2. AN INFEASIBLE INTERIOR POINT (IIP) METHOD FOR SDP **6**

2.1 Introduction 6

2.2 First Order Necessary Condition (FONC) 9

2.3 Predictor Step 11

2.4 Corrector Step 13

2.5 Stopping Conditions 14

3. A HOMOGENEOUS SELF DUAL (HSD) METHOD FOR SDP **16**

3.1 Introduction 16

3.2 Homogeneous self dual linear programming 16

3.3 Homogeneous self dual algorithm for Semidefinite programming 20

3.3.1 First Order Necessary Condition 26

3.3.2 Predictor and Corrector steps 28

3.3.3 Stopping Conditions 37

4. COMPUTATIONAL RESULTS **39**

4.1 Introduction 39

4.2 Comparison of Robustness 42

4.3 Comparison of Efficiency, Iteration Counts and CPU time 42

5. CONCLUSIONS **61**

Bibliography **62**

LIST OF TABLES

4.1 Accuracy (maximum error) 43

4.2 Accuracy (maximum error) 44

4.3 Accuracy (maximum error) 45

4.4 Accuracy (maximum error) 46

4.5 Contingency Table for HSD and IIP 47

4.6 Contingency Table for SeDuMi and CSDP 47

4.7 Time per iteration 48

4.8 Time per iteration 49

4.9 Time per iteration 50

4.10 Time per iteration 51

4.11 Iteration counts 52

4.12 Iteration counts 53

4.13 Iteration counts 54

4.14 Iteration counts 55

4.15 CPU time, seconds 56

4.16 CPU time, seconds 57

4.17 CPU time, seconds 58

4.18 CPU time, seconds 59

CHAPTER 1

LINEAR and SEMIDEFINITE PROGRAMMING

1.1 Linear Programming

Linear programming problems (LP) involve optimizing a linear function of a vector of variables subject to linear constraints. The standard form of the linear programming problems is

$$\begin{aligned} \text{Primal} \qquad & \text{maximize} & c^T x & \qquad (1.1) \\ & \text{subject to} & Ax = b, & \qquad (1.2) \\ & & x \geq 0, & \qquad (1.3) \end{aligned}$$

where c and $x \in R^n, b \in R^m$, and $A \in R^{m \times n}$. Here $c^T x$ is called the objective function; $Ax = b$ is a system of linear equation constraints and $x \geq 0$ is a nonnegativity constraint. If x satisfies the constraints $Ax = b$ and $x \geq 0$, it is called a feasible solution.

The first practically efficient method for linear programming was the simplex method. But as the problems to be solved became larger, its high complexity became an obvious disadvantage, because the simplex method could require an exponential number of iterations. For example, the Klee-Minty problem requires 2^n simplex iterations[15, 22].

In 1979, Khachiyan developed the ellipsoid method which is a polynomial algorithm for linear programming, but because it converges slowly, it is not competitive with the simplex method on big problems[22].

In 1984, Karmarkar developed an interior point method without reference to the dual to solve linear programming that was polynomial, requiring $O(n \log \frac{1}{\varepsilon})$ iterations. It was shown to be practically efficient[22].

In 1992, Yinyu Ye developed a homogeneous self dual algorithm to solve linear programming. It reformulates the linear programming problem and then applies a primal-dual method to solve the reformulated problem[23].

Any linear program has associated with it another linear program that is called the dual and the original one is called the primal. These two problems have the same data objects arranged in different ways. The associated dual problem in standard form is:

$$\text{(Dual)} \qquad \text{minimize} \qquad b^T y \tag{1.4}$$

$$\text{subject to} \qquad A^T y - c \geq 0, \tag{1.5}$$

where $y \in R^m$ is the dual variable.

Assume the constraint matrix A has linearly independent rows. We define the primal-dual feasible set F and strictly feasible set F^0 by :

$F = \{(x, y, z) | Ax = b, A^T y - z = c, x, z \geq 0\}$,

$F^0 = \{(x, y, z) | Ax = b, A^T y - z = c, x, z > 0\}$ [23].

There are several theorems that describe the relationship between the primal and dual problems. Proofs can be found in [5].

Theorem 1 (Weak Duality in Linear Programming). Each value of the primal objective function for a primal feasible solution provides a lower bound for every value of the dual objective function [5].

Theorem 2 (Strong Duality in Linear Programming). If a primal problem possesses an optimal solution then its dual has an optimal solution and optimal values of the two problems are equal [5].

Theorem 3 (Complementary Slackness in Linear Programming). At optimality, each dual linear constraint is active or the corresponding primal variable is 0 or both [5].

We define the duality gap as: the difference between the primal and dual objective values of primal and dual feasible solutions, $b^T y - c^T x$.

1.2 Semidefinite programming

Semidefinite programming is a generalization of linear programming in which symmetric matrices as well as real vectors are included among the variables, and positive semidefiniteness conditions on the matrix variables are included in the constraints. This definition is taken directly from [22].

To extend the idea of Linear Programming to Semidefinite programming, we have semidefinite programming (SDP) problems in the following standard form [3]:

$$\begin{aligned}
&\text{(Primal)} && \text{maximize} && \operatorname{tr}(CX) && (1.6)\\
&&& \text{subject to} && A(X) = b, && (1.7)\\
&&&&& X \succeq 0 && (1.8)\\
&\text{(Dual)} && \text{minimize} && b^T y && (1.9)\\
&&& \text{subject to} && A^T(y) - Z = C, && (1.10)\\
&&&&& Z \succeq 0 && (1.11)
\end{aligned}$$

where

$$A(X) = \begin{bmatrix} \operatorname{tr}(A_1 X) \\ \operatorname{tr}(A_2 X) \\ \vdots \\ \operatorname{tr}(A_m X) \end{bmatrix}$$

$$A^T(y) = \sum_{i=1}^{m} y_i A_i.$$

When $X = \operatorname{diag}(x)$ is a diagonal matrix, the semidefiniteness constraint is simply $x \geq 0$ and the SDP is a linear program. At the same time the objective function $\operatorname{tr}(CX)$ and constraints $A(X) = b$ are linear combination of the elements of X[1, 3, 16].

$X \succeq 0$ is the only nonlinearity in the semidefinite program. The set of symmetric and positive semidefinite matrices forms a convex cone, so we have a convex optimization problem. In practice, X and Z are often block diagonal. Each block must be positive semidefinite[1, 3, 16].

Most primal-dual interior-point methods for linear programming have been generalized to semidefinite programming. As in linear programming,

these methods have polynomial worst-case complexity, and perform very well in practice[22].

Semidefinite programming unifies several standard problems and has been applied to many fields, such as electrical engineering, structural engineering and combinatorial optimization[2, 20].

There are several theorems that describe the relationship between the primal and dual problems to the semidefinite program[6]. Proof can be found in [21].

Theorem 4 (Weak Duality in Semidefinite Programming). Each value of the primal objective function provides a lower bound for every value of the dual objective function [6].

Theorem 5 (Strong Duality in Semidefinite Programming). If the optimal primal and dual objective values are finite and both the primal and dual problems have strictly feasible solutions, then the optimal primal and dual objective values are equal [6].

Theorem 6 (Duality Gap in Semidefinite Programming). If X and Z are primal and dual feasible, the duality gap is equal to $\mathrm{tr}(ZX)$ [6]. When X and Z are primal and dual optimal, tr(ZX)=0. This is the complementary slackness theorem in semidefinite programming.

CHAPTER 2

AN INFEASIBLE INTERIOR POINT (IIP) METHOD FOR SDP

2.1 Introduction

A simple interior-point method for LP starts from a strictly feasible point near the central path, but in some cases it is not easy to find a strictly feasible point. In fact, the set of strictly feasible points is empty for some problems[22]. Infeasible interior-point methods for linear programming do not require the initial point to be strictly feasible, but instead require x^0, z^0 components be strictly positive. Newton's method steps then move successive solutions towards primal and dual feasibility. Infeasible interior point methods have also been developed for SDP[22].

The algorithm used in this project is a predictor corrector variant of the algorithm of Helmberg, Rend, Vanderbei and Wolkowicz (HRVW)[9]. It is known as the HKM method. This is because the method was developed independently in three separate papers. The HKM search direction was developed independently by Helmberg, Rendl, Vanderbei, and Wolkowicz[9], Kojima, Shindoh, and Hara[10] and Monteiro[14]. H, K, M are the initials of the first authors of the three papers. HKM uses Newton's method to solve the first order necessary condition for a logarithmic barrier problem. As the barrier parameter μ is reduced, the optimal solutions to the barrier problem approach optimal solutions to the SDP.

We implement an infeasible interior point method to solve semidefinite programming problems in the MATLAB environment in this project, based on an article of Brian Borchers[1].

We begin with a logarithmic barrier problem[1]:

$$\begin{aligned} &\text{maximize} && \operatorname{tr}(CX) + \mu \log\det X && (2.1)\\ &\text{(PBP)} && A(X) = b && (2.2)\\ & && X \succeq 0 && (2.3) \end{aligned}$$

where $\mu > 0$ is the barrier parameter. To get the dual problem, firstly, we introduce the Lagrange function with the dual variable y:

$$L(y, X) = \operatorname{tr}(CX) + \mu \log\det X - y^T(A(X) - b). \tag{2.4}$$

The dual of this problem will be $\min_y(\max_X(L(y, X)))$. To get the maximum of $L(y, X)$, we take the derivative of L(y,X) respect to X and let it be equal to 0. We obtain

$$C + \mu X^{-1} - A^T(y) = 0. \tag{2.5}$$

This will be the equality constraint of the dual variable problem. Then

$$\mu X^{-1} = A^T(y) - C. \tag{2.6}$$

Let

$$A^T(y) - C = Z, \tag{2.7}$$

so that

$$\mu I = ZX \tag{2.8}$$

or

$$X = \mu Z^{-1}. \tag{2.9}$$

Applying the trace operation to both sides of (2.5), we obtain

$$\mathrm{tr}((C + \mu X^{-1} - A^T(y))X) = 0 \tag{2.10}$$

$$\mathrm{tr}(CX) + \mathrm{tr}(\mu I) - y^T A(X) = 0 \tag{2.11}$$

$$\mathrm{tr}(CX) + n\mu - y^T A(X) = 0 \tag{2.12}$$

$$\mathrm{tr}(CX) - y^T A(X) = -n\mu. \tag{2.13}$$

Rewrite the $\mathrm{tr}(CX) - y^T A(X)$ and X in $L(y, X)$, then we have

$$\max(L(y, Z)) = y^T b - n\mu + \mu \log \det (\mu Z^{-1}) \tag{2.14}$$

$$\max(L(y, Z)) = b^T y - n\mu + n\mu \log \ \mu - \mu \log \det Z. \tag{2.15}$$

In (2.15), "$-n\mu + n\mu \log \ \mu$" is constant, so the dual variable barrier problem can be written as

$$\text{minimize} \qquad b^T y - \mu \log \det Z \tag{2.16}$$

$$\text{(DBP)} \qquad A^T(y) - C = Z \tag{2.17}$$

$$Z \succeq 0. \tag{2.18}$$

The solutions of the barrier problems trace out a curve called the central path. Goldfarb and Scheinberg showed that the primal and dual central path converge to the analytic center of the optimal set[8]. However Halicka, Klerk and Roos showed that this conclusion is correct only when we have a strictly complementary solution[11], which means the solution satisfies rank(X)+rank(Z)=n.

2.2 First Order Necessary Condition (FONC)

Let us get started from the logarithmic barrier problem to derive the FONC. The Lagrangian function of DBP is

$$L(y, Z) = b^T y - \mu \log \det Z - \mathrm{tr}(X^T(A^T(y) - c - Z)). \tag{2.19}$$

Here, y and Z are the dual variables. Taking the derivative of L with respect to Z, we have

$$\frac{\partial L}{\partial Z} = -\frac{\partial(\mu \log \det Z)}{\partial Z} + \frac{\partial(\mathrm{tr}(X^T Z))}{\partial Z} \tag{2.20}$$

$$\frac{\partial L}{\partial Z} = -\mu Z^{-1} + X. \tag{2.21}$$

To get an optimal solution, we need $\partial L/\partial Z = 0$, so $-\mu Z^{-1} + X = 0$. Thus we have

$$\mu I = ZX. \tag{2.22}$$

This is a complementary slackness condition. When we get to the optimal point, μ will be equal to 0. Taking the trace of both sides, we obtain

$$\mathrm{tr}(\mu I) = \mathrm{tr}(ZX) \tag{2.23}$$

$$n\mu = \mathrm{tr}(ZX) \tag{2.24}$$

$$\mu = \frac{\mathrm{tr}(ZX)}{n}, \tag{2.25}$$

as μ approaches 0, tr(ZX) goes to zero.

So a pair of primal and dual optimal solutions to (PBP) and (DBP) satisfy $\mu = \frac{\mathrm{tr}(ZX)}{n}$. Now we have the first order necessary conditions for the

barrier problem

$$Z + C - A^T(y) = 0 \tag{2.26}$$

$$b - A(X) = 0 \tag{2.27}$$

$$\mu I = ZX \tag{2.28}$$

$$X \succeq 0 \tag{2.29}$$

$$Z \succeq 0. \tag{2.30}$$

The first two equations and the last two matrix inequalities come from primal and dual feasibility for the SDP. The third equation is the complementarity condition. Because our initial solutions might not be feasible with respect to the constraints, we define the primal and dual infeasibilities as

$$F_p = b - A(X) \tag{2.31}$$

$$F_d = Z + C - A^T(y). \tag{2.32}$$

If our algorithm strictly follows the central path, i.e., let $\mu = \frac{\mathrm{tr}(ZX)}{n}$ in each iteration, we will find that there is no progress made. On the other hand, if we let $\mu = 0$ in each iteration, the solution quickly reach the boundary. Once it gets close to boundary, it will be very difficult to continue making good progress. So our algorithm consists of two major steps: a predictor step and a corrector step. In the predictor step, we determine how much of a decrease in μ is possible. In the corrector step, we use that μ in system (2.26-2.30) and take a Newton's method step.

Before we can talk about the predictor step and corrector step, we need to know our initialization. We follow the method in K.C.Toh, M.J.Todd

and R.H. Tutuncu's paper[19].

$$X^0 = \text{diag}(\xi_i I_{n_i}), \quad y^0 = 0, \quad Z^0 = \text{diag}(\eta_i I_{n_i}), \tag{2.33}$$

where i=1, $\cdots$, L (number of blocks), n_i is the size of block i, I_{n_i} is the identity matrix of order n_i, and

$$\xi_i = n_i \max \frac{1 + |b_k|}{1 + \|A_k^{(i)}\|_F}, \quad 1 \le k \le m$$

$$\eta_i = \frac{1 + \max[\max_k\{\|A_k^{(i)}\|_F\}, \quad \|C^{(i)}\|_F]}{\sqrt{n_i}}.$$

2.3 Predictor Step

We can apply a Newton's method step for the first order necessary conditions and we let $\mu = 0$. Let

$$\text{F}(X, y, Z) = \begin{pmatrix} Z + C - A^T(y) \\ b - A(X) \\ ZX \end{pmatrix} \tag{2.34}$$

The gradient of F respect to X,y,Z is :

$$\nabla\text{F} = \begin{bmatrix} 0 & -A^T & I \\ -A & 0 & 0 \\ Z & 0 & X \end{bmatrix}. \tag{2.35}$$

Then, the Newton's method step is obtained by solving

$$\nabla\text{F} \begin{bmatrix} \Delta X \\ \Delta y \\ \Delta Z \end{bmatrix} = -\text{F}(X, y, Z). \tag{2.36}$$

By (2.31) and (2.32), we have:

$$\begin{bmatrix} 0 & -A^T & I \\ -A & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta X \\ \Delta y \\ \Delta Z \end{bmatrix} = \begin{bmatrix} -Fd \\ -Fp \\ -ZX \end{bmatrix} \tag{2.37}$$

Expand this system and we have the Newton's method equations for this step

$$-A(\Delta X) = -F_p \tag{2.38}$$

$$\Delta Z - A^T(\Delta y) = -F_d \tag{2.39}$$

$$Z\Delta X + \Delta Z X = -ZX \tag{2.40}$$

.

The following method to solve these three equations is taken directly from a talk of Brian Borchers. We solve for (ΔZ, Δy, ΔX) from these three equations. First, we can solve for $A^T(\Delta\ y)$ from (2.39), we have

$$A^T(\Delta y) = F_d + \Delta Z. \tag{2.41}$$

Then, multiply this equation by Z^{-1} on the left and X on the right and apply $A()$ to both sides and we have

$$A(Z^{-1}A^T(\Delta y)X) = A(Z^{-1}F_d X) + A(Z^{-1}\Delta Z X). \tag{2.42}$$

By equation (2.40), we have $\Delta Z X = -ZX - Z\Delta X$, so

$$A(Z^{-1}A^T(\Delta y)X) = A(Z^{-1}F_d X) + A(Z^{-1}(-ZX - Z\Delta X)) \tag{2.43}$$

$$A(Z^{-1}A^T(\Delta y)X) = A(Z^{-1}F_d X) + A(-X) + A(-\Delta X) \tag{2.44}$$

$$A(Z^{-1}A^T(\Delta y)X) = A(Z^{-1}F_d X) + A(-X) + A(X) - b \tag{2.45}$$

$$A(Z^{-1}A^T(\Delta y)X) = A(Z^{-1}F_d X) - b. \tag{2.46}$$

We define the Schur complement matrix O as

$$O = \left[A(Z^{-1}A_1X)\cdots A(Z^{-1}A_mX)\right]. \tag{2.47}$$

The equation (2.46) can be written as

$$O\Delta y = A(Z^{-1}F_dX) - b. \tag{2.48}$$

Once we've solved for Δy, we can solve for ΔZ and ΔX by

$$\Delta Z = -F_d + A^T(\Delta y) \tag{2.49}$$

$$\Delta X = -X - Z^{-1}\Delta ZX. \tag{2.50}$$

ΔX might not be symmetric, because the product of symmetric matrices could give us an unsymmetric matrix. We can keep ΔX symmetric by letting

$$\Delta X_s = \frac{\Delta X + \Delta X^T}{2}. \tag{2.51}$$

After we get the direction $(\Delta X_s, \Delta y, \Delta Z)$, we need to keep the point feasible. We use a line search to find the maximum step length. Details of the eigenvalue line search technique are in section 3.3.2.

2.4 Corrector Step

When the point is close to the boundary, the primal-dual method performs badly. In the corrector step, we can bring the point back to the central path[16].

We set $\tilde{\mu} = \frac{\mathrm{tr}(Z_pX_p)}{n}$ (the subscript p means we use Z, X computed

from the predictor step), and solve

$$\Delta Z - A^T(\Delta y) = -F_d \tag{2.52}$$

$$-A(\Delta X) = -F_p \tag{2.53}$$

$$Z\Delta X + \Delta Z X = -ZX + \tilde{\mu} I. \tag{2.54}$$

We can solve this system of equations in the same way as the predictor step. The Schur complement matrix O can be reused, so the factorization of O doesn't need to be repeated. After we get the movement direction, we use the same principles as in the predictor step to find the maximum step length. Finally, we update the variables.

2.5 Stopping Conditions

In each iteration, we need to check the following conditions:

Primal Feasibility:

$$\frac{\|A(X) - b\|_2}{1 + \|b\|_2} < \tau$$

Dual Feasibility:

$$\frac{\|A^T(y) - C - Z\|_F}{1 + \|C\|_F} < \tau$$

Duality Gap:

$$\frac{\mathrm{tr}(ZX)}{1 + |b^T y| + |\mathrm{tr}(CX)|} < \tau$$

In theory, for primal and dual feasible solutions,

$$\mathrm{tr}(ZX) = b^T y - \mathrm{tr}(CX).$$

We showed this in section 1.2. In practice, even when the primal and dual feasibility are satisfied, the two sides of this equation can be different because of the computational round off errors. In some cases, the objective function gap is negative. We use $\mathrm{tr}(ZX)$ in this project. We let $\tau =$ 1e-8.

The following are the main steps for IIP:

1. compute the initial value (X^0, y^0, Z^0) by (2.33)
2. use a loop until a stopping criterion is satisfied
   (a) predictor step
      i. compute Schur complement matrix O by (2.47)
      ii. compute the search direction $(\Delta X, \Delta y, \Delta Z)$ using HKM method by (2.50),(2.48),(2.49)
      iii. compute the step length α_p, α_d
      iv. update the variables
   (b) corrector step
      i. compute μ using $\mu = \frac{\mathrm{tr}(Z_p X_p)}{n}$
      ii. compute the search direction (X^0, y^0, Z^0) using HKM method by solving (2.52)(2.53)(2.54)
      iii. compute the step length α_p, α_d
      iv. update the variables

CHAPTER 3

A HOMOGENEOUS SELF DUAL (HSD) METHOD FOR SDP

3.1 Introduction

E. de Klerk showed that the initialization strategy of embedding a linear programming problem in a skew-symmetric self dual problem can also be extended to the semi-definite case[7]. By using this reformulation, the initialization problem of semi-definite problems is solved.

In this project, we are aiming to implement a homogeneous self dual skew-symmetric algorithms to solve semidefinite programming problems in the MATLAB environment. Following the method of SDPHA, a rank-4 update technique is employed in this algorithm[17].

3.2 Homogeneous self dual linear programming

Homogeneous means that the right-hand sides of the constraints are zeros. Self duality can be defined as following[23]:

Let $\tilde{A} \in R^{p \times p}$ be skew-symmetric (meaning $\tilde{A}^T = -\tilde{A}$), and let $\tilde{b} = -\tilde{c} \in R^p$ then the problem

$$\text{minimize} \qquad \tilde{c}^T \tilde{\mu} \tag{3.1}$$

$$\text{subject to} \qquad \tilde{A}\tilde{\mu} \geq \tilde{b}, \tag{3.2}$$

$$\tilde{\mu} \geq 0 \tag{3.3}$$

where $\tilde{\mu} \in R^p$, is called a self dual linear programming problem.

The associated dual of this problem is

$$\begin{aligned} &\text{maximize} && \tilde{b}^T\tilde{y} && (3.4)\\ &\text{subject to} && \tilde{A}^T\tilde{y} \leq \tilde{c}, && (3.5)\\ & && \tilde{y} \geq 0. && (3.6) \end{aligned}$$

Because $\tilde{b} = -\tilde{c}$ and $\tilde{A}^T = -\tilde{A}$, the primal and dual are exactly the same except that the variable names are different. For self dual linear problems, there is the following theorem.

Theorem 7[23]: The self dual linear problem is equivalent to its dual. Suppose that self dual linear problem has a feasible solution $\tilde{\mu}$, then $\tilde{\mu}$ is also feasible in the dual problem, and the two objective values sum to zero. Moreover, in this case self dual linear programming problem has an optimal solution, and its optimal value is zero.

A proof can be found in [23].

Ye presents a homogeneous self dual linear program (HLP) relating (LP) and (LD)[23]. Given any $x^0 \geq 0, z^0 \geq 0$ and y^0, he formulated the HLP as

$$\begin{aligned} &\text{(HLP) min} && ((x^0)^T z^0 + 1)\theta && && (3.7)\\ &\text{subject to} && Ax - b\tau + \bar{b}\theta && = 0 && (3.8)\\ & && -A^T y + c\tau - \bar{c}\theta && \geq 0 && (3.9)\\ & && b^T y - c^T x + \bar{g}^T\theta && \geq 0 && (3.10)\\ & && \bar{b}^T y + \bar{c}^T x - \bar{g}\tau && = -(x^0)^T z^0 - 1 && (3.11) \end{aligned}$$

where y is free, $x \geq 0, \tau \geq 0$, θ is free, $\bar{b} = b - Ax^0$, $\bar{c} = c - A^T y^0 - z^0$, $\bar{z} = c^T x^0 + 1 - b^T y^0$.

Constraints (3.9),(3.10) can be written as

$$-A^T y + c\tau - \bar{c}\theta - z = 0 \tag{3.12}$$

$$b^T y - c^T x + \bar{g}\theta - \kappa = 0 \tag{3.13}$$

where $z \geq 0, \kappa \geq 0$.

There are the following theorems about homogeneous self dual linear problems. They are taken directly from [23].

Theorem 8[23]

1. (HLD) has the same form as (HLP), i.e., (HLD) is simply (HLP) with (y, x, τ, θ) being replaced by (y', x', τ', θ').

2. (HLP) has a strictly feasible point

$$y = y^0, x = x^0, \tau = 1, \theta = 1, z = z^0 > 0, \kappa = 1.$$

3. (HLP) has an optimal solution and its optimal solution set is bounded.

4. The optimal value of (HLP) is zero, and for any feasible point $(y, x, \tau, \theta, z, \kappa) \in F_h$, where F_h denotes the set of all points(y,x,τ,θ,z,κ) that are feasible for (HLP),

$$((x^0)^T z^0)\theta = x^T z + \tau\kappa. \tag{3.14}$$

5. There is an optimal solution $(y^*, x^*, \tau^*, \theta^*, z^*, \kappa^*) \in F_h$ such that

$$\begin{pmatrix} x^* + z^* \\ \tau^* + \kappa^* \end{pmatrix} > 0.$$

Theorem 9[23] Let $(y^*, x^*, \tau^*, \theta^* = 0, z^*, \kappa^*)$ be a strictly self-complementary solution for (HLP).

I. (LP) has a solution (feasible and bounded) if and only if $\tau^* > 0$.In this case, x^*/τ^* is an optimal solution for (LP) and $(y^*/\tau^*, z^*/\tau^*)$ is an optimal solution for (LD).

II. If $\tau^* = 0$ then $\kappa^* > 0$, which implies that $c^T x^* - b^T y^* < 0$, i.e., at least one of $c^T x^*$ and $-b^T y^*$ is strictly less than zero. If $c^T x^* < 0$ then (LD) is infeasible; if $-b^T y^* < 0$ then (LP) is infeasible; and if both $c^T x^*$ and $-b^T y^* < 0$ then both (LP) and (LD) are infeasible.

A proof can be found in [23].

COROLLARY [23]

Let $(\bar{y}, \bar{x}, \bar{\tau}, \bar{\theta} = 0, \bar{z} = 0, \bar{\kappa})$ be any optimal solution for (HLP). Then if $\bar{\kappa} > 0$, either (LP) or (LD) is infeasible.

Interior-point algorithms can be used to solve (HLP). The following theorem resembles the central path analyzed for (LP) and (LD).

Theorem 10[23] For any $\mu > 0$, there is a unique $(y, x, \tau, \theta, z, \kappa)$ in F_h^0 such that

$$\begin{pmatrix} Xz \\ \tau\kappa \end{pmatrix} = \mu e$$

where $X = \text{diag}(x)$ denotes the diagonal matrix with diagonal entries equal to the components of x, and e is an vector with all the entries equal to 1. newheadline Let $(d_y, d_x, d_\tau, d_\theta, d_z, d_\kappa)$ be in the null space Q of the constraint matrix of (HLP) after adding surplus variables z and κ, i.e.,

$$\begin{aligned}
& Ad_x && - bd_\tau && + \bar{b}d_\theta && && = 0 && (3.15)\\
- A^T d_y && && + cd_\tau && - \bar{c}d_\theta && - d_z && = 0 && (3.16)\\
b^T d_y && - c^T d_x && && + \bar{g}^T d_\theta && - d_\kappa && = 0 && (3.17)\\
- \bar{b}^T d_y && + \bar{c}^T d_x && - \bar{g}^T d_\tau && && && = 0 && (3.18)
\end{aligned}$$

Then we have

$$(-d_x)^T d_z - d_\tau d_\kappa = 0. \tag{3.19}$$

Theorem 10 defines the path in (HLP)[23]:

$$L = \left\{ (y, x, \tau, \theta, z, \kappa) \in F_h^0 : \begin{pmatrix} Xz \\ \tau\kappa \end{pmatrix} = \frac{x^T z + \tau\kappa}{n+1} e \right\}. \tag{3.20}$$

3.3 Homogeneous self dual algorithm for Semidefinite programming

In this chapter we use a slightly different form of semidefinite program

$$\begin{aligned}
&\text{(Primal)} && \text{minimize} && \text{tr}(CX) && (3.21)\\
& && \text{subject to} && A(X) = b && (3.22)\\
& && && X \succeq 0 && (3.23)\\
&\text{(Dual)} && \text{maximize} && b^T y && (3.24)\\
& && \text{subject to} && A^T(y) + Z = C && (3.25)\\
& && && Z \succeq 0 && (3.26)
\end{aligned}$$

It can be shown that this form is equivalent to the standard form

$$\begin{aligned}
&\text{(Primal)} && \text{maximize} && \operatorname{tr}(CX) && (3.27)\\
&&& \text{subject to} && A(X) = b && (3.28)\\
&&&&& X \succeq 0 && (3.29)\\
&\text{(Dual)} && \text{minimize} && b^T y && (3.30)\\
&&& \text{subject to} && A^T(y) - Z = C && (3.31)\\
&&&&& Z \succeq 0 && (3.32)
\end{aligned}$$

stated in Chapter1.

Proof: The objective function of the primal problem in standard form (3.19) is equivalent to minimizing $-\operatorname{tr}(CX)$. By multiplying equation(3.22) by -1, we have

$$-A(X) = -b. \tag{3.33}$$

Then, the associated dual will be :

$$\begin{aligned}
&\text{(Dual)} && \text{minimize} && -b^T y && (3.34)\\
&&& \text{subject to} && -A^T(y) + Z = -C && (3.35)\\
&&&&& Z \succeq 0 && (3.36)
\end{aligned}$$

Let $C' = -C$, $b' = -b$, $A' = -A$. Then, the standard form is equivalent to

$$\begin{aligned}
&\text{(Primal)} && \text{minimize} && \operatorname{tr}(C'X) && (3.37)\\
&&& \text{subject to} && A'(X) = b' && (3.38)\\
&&&&& X \succeq 0 && (3.39)\\
&\text{(Dual)} && \text{maximize} && b'^T y && (3.40)\\
&&& \text{subject to} && A'^T(y) + Z = C' && (3.41)\\
&&&&& Z \succeq 0. && (3.42)
\end{aligned}$$

So the first form and the standard form have the same optimal solution, except the objective function values have different signs.

The reason we do this is to match the SDPHA notation. It is easy to transform a standard form of SDP to this form:

1. the primal objective function is changed from

$$\text{maximize} \quad \operatorname{tr}(CX)$$

   to

$$\text{minimize} \quad -\operatorname{tr}(CX).$$

2. the primal constraints are changed from

$$\begin{aligned}
&\text{subject to} && A(X) = b && (3.43)\\
&&& X \succeq 0 && (3.44)
\end{aligned}$$

to

$$\text{subject to} \qquad -A(X) = -b \tag{3.45}$$

$$X \succeq 0. \tag{3.46}$$

3. the dual objective function is changed from

$$\text{minimize} \quad b^T y$$

to

$$\text{maximize} \quad -b^T y.$$

4. the dual constraints are changed from

$$\text{subject to} \qquad A^T(y) - Z = C \tag{3.47}$$

$$Z \succeq 0 \tag{3.48}$$

to

$$\text{subject to} \qquad -A^T(y) + Z = -C \tag{3.49}$$

$$Z \succeq 0. \tag{3.50}$$

Following the HSD method for linear programming, we can rewrite the semi-definite programming problems in this form:

$$\text{primal} \qquad \text{minimize} \qquad c^T x \tag{3.51}$$

$$\text{subject to} \qquad Ax = b \tag{3.52}$$

$$X \succeq 0 \tag{3.53}$$

The dual to primal can be written as

$$\begin{aligned} \text{dual} \quad & \text{maximize} && b^T y && (3.54)\\ & \text{subject to} && A^T y + z = c && (3.55)\\ & && Z \succeq 0 && (3.56) \end{aligned}$$

where $A = [\text{vec}(A_1),\ \text{vec}(A_2), \cdots, \text{vec}(A_m)]^T$, $c = \text{vec}(C)$, $x = \text{vec}(X)$, $z = \text{vec}(Z)$. The notation vec() means the operation of converting a matrix to a column vector with the same elements as the matrix in column-major order. This form is very similar to the form of linear programming.

We can extend the homogeneous self dual algorithm to semidefinite programming. Following the method of the homogeneous self dual linear programming algorithm, the above problems can be embedded in the skew-symmetric self dual problem with nonempty interior as:

$$\begin{array}{llllllll} \min & \bar{\alpha}\theta & & & & & & (3.57)\\ & Ax & -b\tau & +\bar{b}\theta & & & = 0 & (3.58)\\ -A^T y & & +c\tau & -\bar{c}\theta & -z & & = 0 & (3.59)\\ b^T y & -c^T x & & +\bar{g}^T\theta & & -\kappa & = 0 & (3.60)\\ -\bar{b}^T y & +\bar{c}^T x & -\bar{g}\tau & & & & = -\bar{\alpha} & (3.61) \end{array}$$

where

$$A = [\text{vec}(A_1), \text{vec}(A_2), \cdots, \text{vec}(A_m)]^T \tag{3.62}$$

$$c = \text{vec}(C) \tag{3.63}$$

$$x = \text{vec}(X) \tag{3.64}$$

$$z = \text{vec}(Z) \tag{3.65}$$

$$\text{mat}(z) \succeq 0 \tag{3.66}$$

$$\text{mat}(x) \succeq 0 \tag{3.67}$$

$$\tau \geq 0 \tag{3.68}$$

$$\kappa \geq 0 \tag{3.69}$$

$$\bar{b} = \frac{b\tau^0 - Ax^0}{\theta^0} \tag{3.70}$$

$$\bar{c} = \frac{A^T y^0 + z^0 - c\tau^0}{\theta^0} \tag{3.71}$$

$$\bar{g} = \frac{c^T x^0 - b^T y^0 + \kappa^0}{\theta^0} \tag{3.72}$$

$$\bar{\alpha} = \frac{(z^0)^T x^0 + \tau^0 \kappa^0}{\theta^0}. \tag{3.73}$$

It is clear that the above constrains are equivalent to the following form

$$
\begin{array}{llllllll}
\min & \bar{\alpha}\theta & & & & & & (3.74)\\
& A(X) & -b\tau & +\bar{b}\theta & & & = 0 & (3.75)\\
-A^T(y) & & +C\tau & -\bar{C}\theta & -Z & & = 0 & (3.76)\\
b^T y & -\operatorname{tr}(CX) & & +\bar{g}^T\theta & & -\kappa & = 0 & (3.77)\\
-\bar{b}^T y & +\operatorname{tr}(\bar{C}X) & -\bar{g}\tau & & & & = -\bar{\alpha} & (3.78)\\
& & X & \succeq 0 & & & & (3.79)\\
& & Z & \succeq 0 & & & & (3.80)\\
& & \tau & \geq 0 & & & & (3.81)\\
& & \kappa & \geq 0 & & & & (3.82)
\end{array}
$$

where $\bar{C}$ is a matrix converted from the vector $\bar{c}$ with the same elements.

3.3.1 First Order Necessary Condition

To derive the first order necessary conditions, we can use the Lagrangian method. First, we construct the associated barrier problem

$$\min \quad \bar{\alpha}\theta - \mu(\log\det X + \log\det Z + \log\kappa + \log\tau) \tag{3.83}$$

$$
\begin{array}{llllllll}
& +A(X) & -b\tau & +\bar{b}\theta & & & = 0 & (3.84)\\
-A^T(y) & & +C\tau & +\bar{C}\theta & -Z & & = 0 & (3.85)\\
+b^T y & -\operatorname{tr}(CX) & & +\bar{g}^T\theta & & -\kappa & = 0 & (3.86)\\
-\bar{b}^T y & -\operatorname{tr}(\bar{C}X) & -\bar{g}\tau & & & & = -\bar{\alpha}. & (3.87)
\end{array}
$$

where $\mu > 0$ is the barrier parameter. Then we introduce the Lagrangian multipliers $\hat{y} \in R$, $\hat{X} \in R_+^{n \times n}$, $\hat{\tau} \in R_+$ and $\hat{\kappa} \in R_+$. For each $\mu > 0$, there is a corresponding Lagrangian function:

$$L_\mu(X, y, Z, \theta, \kappa, \tau) \tag{3.88}$$

$$= \bar{\alpha}\theta - \mu(\log \det X + \log \det Z + \log \kappa + \log \tau) \tag{3.89}$$

$$- \hat{y}^T(A(X) - b\tau + \bar{b}\theta) \tag{3.90}$$

$$- \operatorname{tr}(\hat{X}^T(-A^T(y) + C\tau + \bar{C}\theta - Z)) \tag{3.91}$$

$$- \hat{\tau}(b^T y + \operatorname{tr}(C^T X) + \bar{g}\theta - \kappa) \tag{3.92}$$

$$- \hat{\theta}(-\bar{b}^T y - \operatorname{tr}(\bar{C}^T X) - \bar{g}\tau + \bar{\alpha}). \tag{3.93}$$

To get the optimal solution, we let $\partial L_\mu / \partial Z = 0$ and $\partial L_\mu / \partial \kappa = 0$.

Then we have $-\mu Z^{-1} + \hat{X} = 0$ and $-\mu\kappa^{-1} + \hat{\tau} = 0$, meaning

$$Z\hat{X} = \mu I \tag{3.94}$$

$$\kappa\hat{\tau} = \mu \tag{3.95}$$

When μ approaches 0, these two equations are also the complementary equations for problem (3.74-3.78). By the properties of the self dual problem (3.74-3.78). At the optimal point, $\hat{X} = X$, $\hat{\tau} = \tau$, we have

$$ZX = 0 \tag{3.96}$$

$$\kappa\tau = 0. \tag{3.97}$$

We can generalize the first order necessary conditions as

$$
\begin{aligned}
& A(X) && - b\tau && + \bar{b}\theta && && = 0 && (3.98)\\
- A^T(y) && && + C\tau && - \bar{C}\theta && - Z && = 0 && (3.99)\\
b^T y && - \operatorname{tr}(CX) && && + \bar{g}^T\theta && - \kappa && = 0 && (3.100)\\
- \bar{b}^T y && - \operatorname{tr}(\bar{C}X) && - \bar{g}\tau && && && = -\bar{\alpha} && (3.101)\\
ZX && && && && && && = 0 && (3.102)\\
\tau\kappa && && && && && && = 0. && (3.103)
\end{aligned}
$$

The first four equations are from constraints and the last two are the complementary equations.

We use the HKM method to obtain a search direction. Let

$$(y^0, X^0, \tau^0, \theta^0, Z^0, \kappa^0) = (0, I, 1, 1, I, 1).$$

It is a strictly feasible solution, because if you plug this point into constraints (3.75-3.78), all of them will be satisfied, also, $X^0 \succ 0$ and $Z^0 \succ 0$.

The algorithm consists of two steps: the predictor step and the corrector step.

3.3.2 Predictor and Corrector steps

Predictor step: The predictor step is simply a Newton's method step for the first order necessary conditions. The equations for this step are

$$
\begin{array}{lllll}
 & A(\Delta X) & -b\Delta\tau & +\bar{b}\Delta\theta & = 0 \quad (3.104) \\
-A^T(\Delta y) & & +C\Delta\tau & -\bar{C}\Delta\theta - \Delta Z & = 0 \quad (3.105) \\
b^T\Delta y & -\mathrm{tr}(C\Delta X) & & +\bar{g}^T\Delta\theta \quad -\Delta\kappa & = 0 \quad (3.106) \\
-\bar{b}^T\Delta y & +\mathrm{tr}(\bar{C}\Delta X) & -\bar{g}\Delta\tau & & = 0 \quad (3.107) \\
Z\Delta X & +\Delta Z X & & & = -ZX \quad (3.108) \\
\Delta\tau\kappa & +\tau\Delta\kappa & & & = -\tau\kappa. \quad (3.109)
\end{array}
$$

We solve for the direction from this equation system. We then compute the proper step length $\bar{\sigma}$ such that $X^k + \bar{\sigma}\Delta X_p > 0$, $Z^k + \bar{\sigma}\Delta Z_p > 0$, $\tau^k + \bar{\sigma}\Delta\tau_p > 0$, $\kappa^k + \bar{\sigma}\bar{\kappa}_p > 0$.

We derived a simple procedure to compute the direction

$$(\Delta y, \Delta X, \Delta\tau, \Delta\theta, \Delta Z, \Delta\kappa)$$

following the method of SDPHA[4].

Let

$$\bar{A}^T = [A^T, -c, -\bar{c}] \tag{3.110}$$

$$r_c = -\tau\kappa. \tag{3.111}$$

Let $E, F \in R^{n^2 \times n^2}$, $R_c \in R^{n \times n}$ be such that

$$E\mathrm{vec}(\Delta X) = \mathrm{vec}(XZ\Delta X) \tag{3.112}$$

$$F\mathrm{vec}(\Delta Z) = \mathrm{vec}(X\Delta ZX) \tag{3.113}$$

$$R_c = -XZX. \tag{3.114}$$

Then equation(3.108) has the equivalent vector form:

$$E\text{vec}(\Delta X) + F\text{vec}(\Delta Z) = \text{vec}(R_c) \tag{3.115}$$

By equation(3.109) and equation(3.111), we have

$$\Delta\kappa = \frac{r_c}{\tau} - \frac{\kappa}{\tau}\Delta\tau. \tag{3.116}$$

By equations(3.105) and (3.110), we have

$$\bar{A}^T \begin{pmatrix} \Delta y \\ \Delta\tau \\ \Delta\theta \end{pmatrix} = -\text{vec}(\Delta Z). \tag{3.117}$$

Multiplying $\bar{A}E^{-1}F$ times both sides of the (3.117), we have

$$\bar{A}E^{-1}F\bar{A}^T \begin{pmatrix} \Delta y \\ \Delta\tau \\ \Delta\theta \end{pmatrix} = -\bar{A}E^{-1}F\text{vec}(\Delta Z). \tag{3.118}$$

By equation(3.115), we have:

$$E^{-1}F\text{vec}(\Delta Z) = -(\text{vec}(\Delta X) - E^{-1}\text{vec}(R_c)) \tag{3.119}$$

Then we have:

$$\bar{A}E^{-1}F\bar{A}^T \begin{pmatrix} \Delta y \\ \Delta\tau \\ \Delta\theta \end{pmatrix} = \bar{A}(\text{vec}(\Delta X) - E^{-1}\text{vec}(R_c)) \tag{3.120}$$

$$\bar{A}E^{-1}F\bar{A}^T \begin{pmatrix} \Delta y \\ \Delta\tau \\ \Delta\theta \end{pmatrix} = \bar{A}\text{vec}(\Delta X) - \bar{A}E^{-1}\text{vec}(R_c). \tag{3.121}$$

Because

$$\bar{A}^T = [A^T, -c, -\bar{c}], \tag{3.122}$$

and

$$r_c = -\tau\kappa, \tag{3.123}$$

$$\bar{A}\text{vec}(\Delta X) - \bar{A}E^{-1}\text{vec}(R_c) = \begin{pmatrix} A\text{vec}(\Delta X) \\ -c^T\text{vec}(\Delta X) \\ -\bar{c}^T\text{vec}(\Delta X) \end{pmatrix} - \bar{A}E^{-1}\text{vec}(R_c). \tag{3.124}$$

By equations(3.104) (3.106) and (3.107), we have

$$\bar{A}\text{vec}(\Delta X) - \bar{A}E^{-1}\text{vec}(R_c) = \begin{pmatrix} \Delta\tau b - \Delta\theta\bar{b} \\ -b^T\Delta y - \bar{g}\Delta\theta + \Delta\kappa \\ \bar{b}^T\Delta y + \bar{g}\Delta\tau \end{pmatrix} - \bar{A}E^{-1}\text{vec}(R_c). \tag{3.125}$$

Because $\Delta\kappa = \frac{r_c}{\tau} - \frac{\kappa}{\tau}\Delta\tau$, we have

$$\bar{A}\text{vec}(\Delta X) - \bar{A}E^{-1}\text{vec}(R_c) = \begin{pmatrix} \Delta\tau b - \Delta\theta\bar{b} \\ -b^T\Delta y - \bar{g}\Delta\theta + \frac{r_c}{\tau} - \frac{\kappa}{\tau}\Delta\tau \\ \bar{b}^T\Delta y + \bar{g}\Delta\tau \end{pmatrix} - \bar{A}E^{-1}\text{vec}(R_c). \tag{3.126}$$

$$\bar{A}\text{vec}(\Delta X) - \bar{A}E^{-1}\text{vec}(R_c) = \begin{pmatrix} 0_{m\times m} & b & -\bar{b} \\ -b^T & -\frac{\kappa}{\tau} & -\bar{g} \\ \bar{b}^T & \bar{g} & 0 \end{pmatrix} \begin{pmatrix} \Delta y \\ \Delta\tau \\ \Delta\theta \end{pmatrix}$$

$$+ \begin{pmatrix} 0_{m\times 1} \\ \frac{r_c}{\tau} \\ 0 \end{pmatrix} - \bar{A}E^{-1}\text{vec}(R_c). \tag{3.127}$$

Let

$$r_h = \begin{pmatrix} 0_{m\times 1} \\ \frac{r_c}{\tau} \\ 0 \end{pmatrix} - \bar{A}E^{-1}vec(R_c). \tag{3.128}$$

By (3.121), (3.127) we have

$$\left[\bar{A}E^{-1}F\bar{A}^T - \begin{pmatrix} 0_{m\times m} & b & -\bar{b} \\ -b^T & -\frac{\kappa}{\tau} & -\bar{g} \\ \bar{b}^T & \bar{g} & 0 \end{pmatrix}\right] \begin{pmatrix} \Delta y \\ \Delta\tau \\ \Delta\theta \end{pmatrix} = r_h. \tag{3.129}$$

Let

$$\mathrm{O} = \bar{A}E^{-1}F\bar{A}^T \tag{3.130}$$

$$= \left[\bar{A}(Z^{-1}\bar{A}_1X)\cdots\bar{A}(Z^{-1}\bar{A}_mX)\bar{A}(Z^{-1}(-C)X)\bar{A}(Z^{-1}(-\bar{C})X)\right]. \tag{3.131}$$

We notice that the computation of the matrix O is the same as in the infeasible interior method, except that we use $\bar{A}$ instead of A. Let

$$M = \mathrm{O} + \begin{pmatrix} 0_{m\times m} & 0_{m\times 1} & 0_{m\times 1} \\ 0_{1\times m} & \frac{\kappa}{\tau}+\alpha_1 & 0 \\ 0_{1\times m} & 0 & \alpha_2 \end{pmatrix} \tag{3.132}$$

$$f_1 = \begin{pmatrix} 0^m \\ 1 \\ 0 \end{pmatrix}, g_1 = \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix}, f_2 = \begin{pmatrix} 0^m \\ 0 \\ -1 \end{pmatrix}, g_2 = \begin{pmatrix} \bar{b} \\ \bar{g} \\ 0 \end{pmatrix} \tag{3.133}$$

$$p = (f_1, f_2, -\alpha_1 f_1 - g_1, -\alpha_1 f_2 - g_2); q = (g_1, g_2, f_1, f_2). \tag{3.134}$$

Equation(3.129) can be written as:

$$\left(M + pq^T \right) \begin{pmatrix} \Delta y \\ \Delta\tau \\ \Delta\theta \end{pmatrix} = r_h \tag{3.135}$$

It can be shown that M is symmetric and positive definite following the method in [9].

Proof: If we can show O is symmetric and positive definite, we can say that M is symmetric and positive definite. By definition O=$\bar{A}E^{-1}F\bar{A}^T$.

$$\text{Let} \quad \bar{A}(v) = v_1A_1 + v_2A_2 + \cdots + v_{m+1}(-C) + v_{m+2}(-\bar{C}).$$

For any $v \in R^{m+2}$,

$$\begin{aligned}
\text{Let} \quad N &= v^T(O)v \\
&= v^T(\bar{A}E^{-1}F\bar{A}^T)v \\
&= (\bar{A}^T v)^T E^{-1} F \bar{A}^T v \\
&= \text{vec}(\bar{A}^T(v))^T E^{-1} F \text{vec}(\bar{A}^T(v)) \\
&= (\text{vec}(\bar{A}^T(v)))^T \text{vec}(X\bar{A}^T(v)Z^{-1}) \\
&= \text{tr}(\bar{A}^T(v)^T X \bar{A}^T(v) Z^{-1}) \\
&= \text{tr}(\bar{A}^T(v) X^{\frac{1}{2}} X^{\frac{1}{2}} \bar{A}^T(v) Z^{-\frac{1}{2}} Z^{-\frac{1}{2}}) \\
&= \text{tr}(Z^{-\frac{1}{2}} \bar{A}^T(v) X^{\frac{1}{2}} X^{\frac{1}{2}} \bar{A}^T(v) Z^{-\frac{1}{2}}) \\
&= (\text{vec}(Z^{-\frac{1}{2}} \bar{A}^T(v) X^{\frac{1}{2}}))^T \text{vec}((X^{\frac{1}{2}} \bar{A}^T(v) Z^{-\frac{1}{2}})^T) \\
&= (\text{vec}(Z^{-\frac{1}{2}} \bar{A}^T(v) X^{\frac{1}{2}}))^T \text{vec}(Z^{-\frac{1}{2}} \bar{A}^T(v) X^{\frac{1}{2}}) \\
&= k^T k.
\end{aligned}$$

Since X and Z^{-1} are positive definite and the equality constraints are linearly independent, we have $k = \text{vec}(Z^{-\frac{1}{2}} \bar{A}^T X^{\frac{1}{2}}) = 0$, if and only if v=0. When $v \neq 0$, we have $N > 0$. It follows that the matrix O is positive definite.

Because A_i, X, A_j, Z^{-1} are symmetric, we have

$$\begin{aligned}
&O_{ij}\\
=&\mathrm{tr}(A_i Z^{-1} A_j X)\\
=&\mathrm{tr}((A_i Z^{-1} A_j X)^T)\\
=&\mathrm{tr}(X^T A_j^T (Z^{-1})^T A_i^T)\\
=&\mathrm{tr}(X A_j Z^{-1} A_i)\\
=&\mathrm{tr}(A_j Z^{-1} A_i X)\\
=&O_{ji}.
\end{aligned}$$

It follows that matrix O is symmetric. At the same time, $\frac{\kappa}{\tau} + \alpha_1 \geq 0$, $\alpha_2 \geq 0$ and

$$M = \mathrm{O} + \begin{pmatrix} 0_{m\times m} & 0_{m\times 1} & 0_{m\times 1} \\ 0_{1\times m} & \frac{\kappa}{\tau} + \alpha_1 & 0 \\ 0_{1\times m} & 0 & \alpha_2 \end{pmatrix} \tag{3.136}$$

Thus M is symmetric and positive definite.

So we can let $M = LL^T$. Then by the last equation, we have

$$\begin{pmatrix} \Delta y \\ \Delta\tau \\ \Delta\theta \end{pmatrix} = (M + pq^T)^{-1} r_h \tag{3.137}$$

$$= (LL^T + pq^T)^{-1} r_h \tag{3.138}$$

$$= [L[I + (L^{-1}p)(L^{-1}q)^T]L^T]^{-1} r_h. \tag{3.139}$$

By using the fact that[15]

$$(I + \mu v^T)^{-1} = I - \mu(I_4 + v^T\mu)^{-1} v^T, \tag{3.140}$$

we have

$$\begin{pmatrix} \Delta y \\ \Delta\tau \\ \Delta\theta \end{pmatrix} = L^{-T}[I - \mu(I_4 + v^T u)^{-1} v^T]w \tag{3.141}$$

$$= L^{-T}[w^T - \mu(I_4 + v^T\mu)^{-1} v^T w] \tag{3.142}$$

where $\mu = L^{-1}p, v = L^{-1}q, w = L^{-1}r_h$. Using the FONC and we can now solve for $\Delta X, \Delta Z$ and $\Delta\kappa$. Finally, compute the step length using the eigenvalue technique stated in the next page.

Corrector Step: After we get the step direction $(\Delta y_p, \Delta X_p, \Delta\tau_p, \Delta\theta_p, \Delta Z_p, \Delta\kappa_p)$ in the predictor step, we solve this following linear system in the corrector step

$$A(\Delta X) - b\Delta\tau \quad + \bar{b}\Delta\theta \quad = 0 \tag{3.143}$$

$$-A^T(\Delta y) \quad + C\Delta\tau \quad - \bar{C}\Delta\theta - \Delta Z \quad = 0 \tag{3.144}$$

$$+b^T\Delta y \quad -\mathrm{tr}(C\Delta X) \quad + \bar{g}^T\Delta\theta \quad -\Delta\kappa = 0 \tag{3.145}$$

$$-\bar{b}^T\Delta y \quad +\mathrm{tr}(\bar{C}\Delta X) - \bar{g}\Delta\tau \quad = 0 \tag{3.146}$$

$$Z\Delta X + \Delta Z X = (\xi\mu I - ZX) - \Delta Z_p \Delta X_p \tag{3.147}$$

$$\Delta\tau\kappa + \tau\Delta\kappa = \xi\mu - \tau\kappa - \Delta\tau_p\Delta\kappa_p \tag{3.148}$$

where

$$\mu = \frac{\mathrm{tr}(Z_p X_p) + \tau_p \times \kappa_p}{n+1} \tag{3.149}$$

and $\xi = 1$. Then we compute the proper step length $\bar{\sigma}$ such that $X^k + \bar{\sigma}\Delta X_c > 0, Z^k + \bar{\sigma}\Delta Z_c > 0, \tau^k + \bar{\sigma}\Delta\tau_c > 0, \kappa^k + \bar{\sigma}\Delta\kappa_c > 0$.

We followed the method of SDPHA to compute the step length $\bar{\sigma}$[4].

Let

$$X = LL^T. \tag{3.150}$$

Because we need to keep $X + \bar{\sigma}\Delta X \succeq 0$, we need

$$L^{-1}(X + \bar{\sigma}\Delta X)L^{-T} \succeq 0 \tag{3.151}$$

or

$$I + \bar{\sigma}L^{-1}\Delta X L^{-T} \succeq 0 \tag{3.152}$$

or

$$\bar{\sigma}L^{-1}\Delta X L^{-T} \succeq -I. \tag{3.153}$$

i.e.

$$L^{-1}\Delta X L^{-T} \succeq -(\frac{1}{\bar{\sigma}})I \tag{3.154}$$

i.e.

$$-\min(\text{eig}(L^{-1}\Delta X L^{-T})) \leq \frac{1}{\bar{\sigma}} \tag{3.155}$$

where min eig means to find the minimum eigenvalue of the matrix.

$$\text{Let} \quad \sigma_1 = -\min(\text{eig}(L^{-1}\Delta X L^{-T})) \tag{3.156}$$

We use the same principle for Z to compute the step length.

$$\text{Let} \quad \sigma_2 = -\min(\text{eig}(L^{-1}\Delta Z L^{-T})). \tag{3.157}$$

$$\text{Let} \quad \sigma_3 = -\Delta\tau/\tau. \tag{3.158}$$

$$\text{Let} \quad \sigma_4 = -\Delta\kappa/\kappa. \tag{3.159}$$

Then $\bar{\sigma} = \frac{1}{max(\sigma_1,\sigma_2,\sigma_3,\sigma_4)+100\epsilon}$, where ϵ is the machine epsilon. Finally, update the variables.

3.3.3 Stopping Conditions

When the following conditions are met, we stop computation.

Primal Feasibility:

$$\frac{\|\frac{\tau b - A^T \mathrm{vec}(X)}{\tau}\|}{\|b\| + 1} < \mathrm{tol}$$

Dual Feasibility:

$$\frac{\| - \sum_{i=1}^{m} y_i A + C_i \tau - Z\|_F}{\|C\|_F + 1} < \mathrm{tol}$$

Duality Gap:

$$\frac{\mathrm{tr}(ZX)}{\max\{1, (b^T y + \mathrm{tr}(CX))\}} < \mathrm{tol}$$

In the code, we let tol=1e-8. The solution of the original problem can be calculated by

$$X_{original} = X/\tau$$

$$y_{original} = y/\tau$$

$$Z_{original} = Z/\tau.$$

In the case that the original problem is infeasible, τ approaches 0 as the number of iteration approaches infinity[4].

The infeasibility criterion in our code is :

$$\frac{\tau}{\kappa + 1} < \text{infeasibility tol.}$$

This condition should keep τ not too small and κ not too large[4]. We use 1e-5 as the infeasibility tolerance.

The following are the main steps for HSD:

1. Let (X^0, y^0, Z^0, τ, θ, κ) =(I,0,I,1,1,1)

2. use a loop until the stopping criterion is satisfied

   (a) predictor step

      i. compute the Schur complement matrix O by (3.131)

      ii. compute the search direction (ΔX, Δy, ΔZ) using HKM method by (3.142),(3.105),(3.108),(3.109)

      iii. compute the step length α_p, α_d

      iv. update the variables

   (b) corrector step

      i. compute μ using (3.149)

      ii. compute the search direction (ΔX, Δy, ΔZ) using the HKM method by (3.143)-(3.148)

      iii. compute the step length α_p, α_d

      iv. update the variables

CHAPTER 4

COMPUTATIONAL RESULTS

4.1 Introduction

In our project, we implement the IIP and HSD algorithms.

We can see IIP and HSD are comparable because they follow the same procedure, use the same technique to compute the Schur matrix O, use the same method to compute the step length and they are implemented in the same environment

In this chapter we will compare these two algorithms in terms of numerical results. We got the test problems from [2] and [12].

All of the test problems were solved on a computer with a 2.8GHz P4 processor and 4GB of RAM, under Linux.

There are 137 problems in the test pool. The range of problem sizes: for the number of constraints matrix m, the range is (6,12376); n is the size of constraints matrix, the range is (13,102606). We didn't run all of the test problems from the source [2, 12], because some of them are too large to be solved on our computer. We set the time limit as 12 hours and memory limit as 4GB.

Let us discuss the problems which were solved with IIP, HSD, CSDP and SeDuMi. CSDP and SeDuMi are two successful implementations for SDP[1, 18].

The **arch** set are truss topology design problems[2]. HSD can solve the first three problems more efficiently compared to IIP. CSDP and SeDuMi can also solve them taking less time. The iteration count for HSD is around half of IIP which is consistent with the general rule.

The **control** set is a series of control theory problems[2]. CSDP can solve all of them, all the other three methods failed in some of them.

The **copos** set are for checking a sufficient condition for copositivity of a matrix[12]. All the four methods can solve them.

The **cphil** problem is of determining copositivity of the Hilbert matrix[12]. All of them can solve cphil10, but for cphil12 SeDuMi and HSD failed.

The **hamming** set are the problems which compute the theta function of Hamming graphs for which the exact value is known[12]. Both HSD and IIP performed excellently. CSDP and SeDuMi can solve them also.

The **mcp** set is a series of max cut problems[2]. All the four methods can solve them very well.

The **theta** set expresses a group of Lovasz theta problems[2]. From theta1 to theta6, all four methods can solve them. But for thetaG11, HSD and IIP failed; for thetaG51, HSD can solve it and IIP failed because of the time limit.

The **torus** set is max cut problems from the Ising model of spin glasses[12]. All four methods could solve torusg3-8 and torusg3-8-50. But only CSDP can solve torusg3-15 and torusg3-15-50.

The **truss** problems outline an assortment of truss topology design

problems[2].

The **yalsdp** problem is YALMIP SDP example with n=100[12]. Except IIP, all the other three methods can solve it.

CSDP and SeDuMi perform much better than HSD and IIP. CSDP can handle some huge problems. Generally memory is not a problem. HSD and IIP couldn't solve these huge problems because of the memory required. Some other problems are not very big, but they are very dense. In this case, HSD and IIP couldn't solve them because of the time limit.

The performance measures include robustness, iteration counts and CPU time. A collection of six error measures were adopted for the DIMACS Challenge on semidefinite programming [13]. We will use the maximum of the six DIMACS errors as a measure of the accuracy. In our notation:

$$\begin{aligned}
\text{err1} &= \frac{\|A(X)-b\|_2}{1+\|b\|_\infty} \\
\text{err2} &= \max\left(0, \frac{-\lambda_{min}(X)}{1+\|b\|_\infty}\right) \\
\text{err3} &= \frac{\|A^T(y)-C-Z\|_F}{1+\|\text{vec}(C)\|_\infty} \\
\text{err4} &= \max\left(0, \frac{-\lambda_{min}(Z)}{1+\|\text{vec}(C)\|_\infty}\right) \\
\text{err5} &= \frac{b^Ty-\text{tr}(CX)}{1+|\text{tr}(CX)|+|b^Ty|} \\
\text{err6} &= \frac{\text{tr}(ZX)}{1+|\text{tr}(CX)|+|b^Ty|}
\end{aligned}$$

If all the six error are $<$ 1.0E-6, we can say this problem is solved. We will count the number of correctly solved problems as a measure of robustness.

The test problems can be downloaded from [2] and [12].

4.2 Comparison of Robustness

HSD method solved 67 problems out of 137; IIP solved 57 problems out of 137; SeDuMi solved 102 problems out of 137; CSDP solved 125 problems out of 137 when the tolerance was equal to 1E-8.

To compare the robustness between HSD and IIP, we can do the Fisher's exact test. The P-value(2-tail) was 0.28 and because it was greater than 0.05, we can say there is no significant difference between IIP and HSD.

To compare the robustness between SeDuMi and CSDP, we can do the Fisher's exact test. The P-value(2-tail) was 0.0003 and because it was less than 0.05, we can say there is significant difference between SeDuMi and CSDP.

Obviously, CSDP is the best one and SeDuMi is also good and both perform better than HSD and IIP.

4.3 Comparison of Efficiency, Iteration Counts and CPU time

The following are the main steps to compute the geometric mean and 95% confidence interval:

1. Compute the ratio of the two sets of data.
2. Take the logs (to the base 10) of the ratio.
3. Compute a conventional mean and 95% confidence interval based on $\bar{X} \pm \frac{s}{\sqrt{n}}$ for the logs. Call the mean M, the lower limit LL, and the upper limit UL.
4. Raise 10 to the powers LL, M, and UL to get a 95% CI around the geometric mean. Due to the laws of logarithms, 10^M will be the geometric mean of the original data.

problem	SEDUMI	CSDP	HSD	IIP
tol	1E-8	1E-8	1E-8	1E-8
arch0	4.60E-10	1.63E-09	4.42E-08	3.98E-09
arch2	1.02E-10	6.75E-09	1.84E-08	1.07E-09
arch4	1.89E-09	5.12E-09	6.27E-08	5.76E-09
arch8	3.72E-10	2.83E-09	4.37E-08	8.80E-09
BH2_r14	3.37E-09	3.82E-09	3.17E-06*	2.21E-07
biggs	3.72E-07	7.10E-08	3.31E-09	1.16E+05*
bm1	2.52E-08	9.06E-07	1.37E-05*	9.69E-01*
buck3	6.50E-08	8.30E-07	6.36E-06*	9.45E-07
buck4	9.93E-10	5.78E-07	1.60E-05*	8.93E-04*
buck5	M	3.30E-06*	T	T
butcher	M	1.74E-07	M	M
C._1.r14	M	2.14E-09	5.94E-07	5.60E-01*
C._3.r14	1.24E-08	3.98E-09	1.69E-06*	1.30E+01*
cancer_100	M	3.62E-07	M	M
checker_1.5	M	9.43E-09	M	M
cnhil10	1.07E-08	2.54E-08	1.04E-07	9.53E-01*
cnhil8	1.41E-08	1.97E-08	1.16E-07	2.66E-01*
control1	2.41E-09	1.66E-09	1.66E-07	5.36E-09
control10	3.26E-06*	1.35E-07	1.28E-06*	1.40E-06*
control11	3.11E-06*	8.64E-08	2.16E-06*	2.18E-06*
control2	3.61E-09	2.89E-09	9.99E-07	1.82E-07
control3	5.54E-08	1.93E-08	5.43E-06*	2.15E-07
control4	3.21E-08	6.58E-09	4.59E-07	1.37E-07
control5	1.07E-06*	3.77E-08	2.81E-06*	1.27E-06*
control6	2.58E-06*	1.05E-07	1.59E-06*	1.41E-06*
control7	9.10E-07	6.81E-08	7.69E-07	5.40E-07
control8	2.07E-06*	3.34E-08	1.20E-05*	5.10E-07
control9	6.00E-07	4.47E-08	1.23E-05*	8.67E-07
copo14	5.93E-09	3.37E-08	1.53E-08	7.54E-09
copo23	2.75E-08	5.62E-07	1.05E-08	5.93E+01*
cphil10	5.40E-15	5.78E-10	2.66E-09	2.81E-10
cphil12	M	6.32E-10	T	4.67E-10
EqualG11	3.10E-11	1.03E-07	7.63E-06*	1.41E-07
EqualG51	9.11E-11	1.46E-07	1.86E-05*	9.74E-09
ex2_1_5	M	1.85E-08	T	T

Table 4.1: Accuracy (maximum error)

problem tol	SEDUMI 1E-8	CSDP 1E-8	HSD 1E-8	IIP 1E-8
filter48_socp	2.29E-09	6.92E-09	7.71E-07	8.24E-04*
foot	2.11E-02*	2.98E-07	T	M
G40_mb	M	9.13E-09	4.01E+02*	M
G40mc	8.46E-10	8.32E-09	1.77E+00*	M
gpp100	2.80E-08	3.68E-08	2.20E-07	3.41E-03*
gpp124-1	6.24E-08	2.57E-08	1.93E-06*	2.28E-01*
gpp124-2	1.85E-08	2.00E-08	9.19E-06*	1.09E-02*
gpp124-3	1.89E-08	2.89E-08	2.02E-06*	8.48E-04*
gpp124-4	2.00E-08	2.16E-08	5.12E-07	1.02E-03*
gpp250-1	4.73E-08	3.61E-08	3.26E-06*	1.87E-01*
gpp250-2	6.02E-08	7.80E-08	1.22E-06*	8.56E-03*
gpp250-3	3.93E-07	6.42E-08	2.83E-06*	7.52E-04*
gpp250-4	7.93E-08	7.59E-08	2.21E-06*	2.83E-04*
gpp500-1	7.66E-08	1.65E-08	1.02E-06*	3.70E-01*
gpp500-2	6.95E-08	1.03E-07	3.38E-06*	4.81E-02*
gpp500-3	2.48E-08	5.09E-08	1.02E-05*	2.83E-03*
gpp500-4	5.98E-08	9.78E-08	7.10E-06*	1.84E-03*
H2O_r14g	M	6.49E-09	1.34E-07	1.48E-05*
H2O+_r14	M	3.68E-09	9.22E-08	M
H3O+_r16	1.88E-08	1.36E-09	3.30E-02	M
hamming_7_5_6	5.58E-11	1.76E-07	2.77E-09	4.88E-09
hamming_9_8	9.36E-09	8.87E-07	6.22E-09	2.38E-09
hand	3.38E-07	2.47E-08	3.03E-03*	M
inc_1200	M	5.67E-03*	1.86E+02*	M
inc_600	2.35E-08	2.79E-04*	1.08E-04*	9.99E-01*
mater_3	3.66E-09	3.75E-09	6.59E-02*	8.92E-09
mater_4	4.52E-09	1.12E-08	T	M
mater_5	3.14E-09	3.00E-09	T	M
maxG11	9.29E-10	2.02E-09	M	5.07E-09
maxG32	4.36E-10	1.23E-08	3.36E-01*	M
maxG51	7.32E-10	2.32E-09	T	M
mcp100	4.13E-10	1.05E-08	1.45E-08	7.75E-09
mcp124-1	1.42E-09	6.12E-09	7.26E-09	5.88E-09
mcp124-2	1.10E-09	4.69E-08	2.53E-09	1.03E-09
mcp124-3	1.21E-09	8.00E-09	5.45E-09	1.52E-09

Table 4.2: Accuracy (maximum error)

problem tol	SEDUMI 1E-8	CSDP 1E-8	HSD 1E-8	IIP 1E-8
mcp124-4	5.33E-10	6.35E-09	4.84E-09	8.87E-09
mcp250-1	5.87E-10	1.09E-08	3.38E-09	9.22E-09
mcp250-2	9.24E-10	1.10E-08	7.58E-09	2.69E-09
mcp250-3	9.78E-10	1.17E-08	1.01E-08	9.41E-09
mcp250-4	1.33E-09	3.27E-08	6.76E-09	4.65E-09
mcp500-1	4.04E-10	3.24E-09	3.39E-08	1.35E-09
mcp500-2	5.83E-10	2.21E-09	1.10E-08	7.52E-09
mcp500-3	5.35E-10	7.70E-09	1.02E-08	4.19E-09
mcp500-4	9.69E-10	2.82E-08	7.20E-08	3.25E-09
neosfbr20	M	4.87E-09	4.63E-08	T
neu1	9.71E-05*	9.20E-04*	1.67E-06	1.69E+04*
neu1g	3.43E-08	5.32E-08	7.79E-09	1.55E-05*
neu2	3.47E-06*	8.34E-10	2.65E-09	2.04E+04*
neu2c	2.22E-06*	9.16E-05*	1.92E-06*	1.22E+03*
neu2g	9.00E-07	7.20E-09	1.20E-08	7.65E-08
neu3	2.76E-08	6.30E-04*	6.97E-07	1.85E+08*
neu3g	M	6.87E-09	1.87E-07	1.00E+00*
NH2_r14	M	4.72E-09	6.63E-08	1.24E+01*
NH3_1-r16	2.45E-08	1.74E-09	4.98E-02*	9.93E-01*
qap10	1.47E-05*	5.18E-08	4.96E-05*	8.92E-07
qap5	6.55E-09	2.08E-08	9.35E-10	1.34E-09
qap6	1.53E-05*	6.92E-09	1.93E-06*	1.59E-06*
qap7	1.71E-05*	9.96E-09	3.50E-06*	2.85E-07
qap8	1.01E-05*	3.84E-08	6.55E-06*	1.54E-06*
qap9	9.87E-06*	4.15E-08	1.25E-05*	3.90E-07
qpG11	2.32E-09	4.09E-08	1.81E-07	7.34E-09
qpG51	5.66E-10	1.82E-08	2.76E-08	4.48E-01*
rabmo	2.67E-10	8.92E-09	T	T
reimer5	M	9.53E-08	T	T
rendl1_600_0	1.08E-09	1.12E-07	3.82E-05*	1.62E-04*
rose13	2.28E-06*	3.76E-07	6.07E-07	1.05E-03*
rose15	4.54E-05*	7.59E-07	1.03E-07	1.00E+00*
shmup3	4.05E-11	2.54E-08	8.40E-07	5.82E-02*
shmup4	1.12E-09	7.75E-07	M	2.23E+06*
shmup5	M	2.05E-05*	M	M

Table 4.3: Accuracy (maximum error)

problem	SEDUMI	CSDP	HSD	IIP
tol	1E-8	1E-8	1E-8	1E-8
ss30	3.47E-08	6.65E-09	7.19E-06*	5.17E-09
st_jcbpaf2	5.18E-10	3.42E-08	M	M
swissroll	2.43E-02*	2.69E-02*	8.83E-02*	1.13E+01*
taha1a	1.08E-09	9.43E-09	1.20E-08	2.14E+02*
taha1b	M	7.47E-09	4.53E-09	M
theta1	1.01E-09	1.00E-07	1.89E-09	5.83E-09
theta2	1.46E-09	7.15E-09	1.01E-08	5.55E-09
theta3	3.51E-09	1.72E-08	3.88E-09	2.98E-09
theta4	1.06E-09	8.13E-09	6.80E-09	7.21E-10
theta5	3.13E-09	2.75E-08	2.00E-09	9.54E-09
theta6	4.18E-09	2.30E-07	3.12E-08	5.82E-09
thetaG11	1.37E-10	7.35E-09	9.10E+00*	2.94E-02*
thetaG51	M	4.87E-09	9.34E-08	T
torusg3_15	M	3.19E-08	M	T
torusg3_8	7.63E-10	2.39E-08	9.34E-08	1.68E-09
toruspm3_15_50	M	1.44E-09	M	T
toruspm3_8_50	4.70E-10	1.86E-08	6.34E-09	2.64E-09
trto3	1.10E-09	1.19E-07	5.50E-06*	1.49E-06*
trto4	1.98E-08	6.58E-06*	3.31E-05*	2.24E+04*
trto5	1.99E-05*	1.08E-05*	4.44E+03*	T
truss1	3.71E-09	5.16E-10	8.95E-10	5.45E-09
truss2	2.49E-09	6.06E-10	2.44E-07	8.10E-09
truss3	2.38E-09	9.34E-10	7.17E-09	1.10E-09
truss4	5.69E-09	5.76E-10	5.75E-09	9.32E-10
truss5	2.97E-09	2.00E-09	6.52E-09	3.05E-09
truss6	5.84E-09	1.01E-08	8.33E-07	4.04E-07
truss7	6.00E-09	2.46E-08	8.64E-07	3.50E-07
truss8	2.22E-09	1.73E-09	2.16E-09	2.99E-09
vibra3	1.47E-09	1.89E-06*	4.66E-06*	1.38E-06*
vibra4	1.80E-09	1.35E-06*	1.61E-06*	2.43E-01*
vibra5	2.31E+00*	5.57E-06*	3.78E+03*	1.52E+06*
yalsdp	1.72E-08	1.87E-08	6.51E-09	1.57E+00*

Table 4.4: Accuracy (maximum error)

*:the max error > 1e-6
M: run out of memory
T: run out of time

	solved	failed
IIP	57	80
HSD	67	70

Table 4.5: Contingency Table for HSD and IIP

	solved	failed
SeDuMi	102	35
CSDP	125	12

Table 4.6: Contingency Table for SeDuMi and CSDP

HSD vs IIP:

In terms of time per iteration: The geometric mean of HSD/IIP is 1.54 and the confidence interval is (1.33,1.78). There is statistically significant difference between them.

In terms of iteration counts (these results were computed only for problems solved by both HSD and IIP): The geometric mean of HSD/IIP is 0.55 and the confidence interval is (0.50,0.61). Obviously, HSD used fewer iterations.

In terms of total computation time (these results were computed only for problems solved by both HSD and IIP): The geometric mean of HSD/IIP is 0.82 and the confidence interval is (0.64,0.96). HSD is a little faster than IIP.

Now let us look at the efficiency. We compare all the others to CSDP. (CSDP is written in C language; SeDuMi, HSD and IIP are both written in Matlab and C).

IIP vs CSDP (these results were computed only for problems solved by both IIP and CSDP): The geometric mean of IIP(time)/CSDP(time) is 23.9 and the confidence interval is (16.89,33.81).

HSD vs CSDP (these results were computed only for problems solved by both HSD and CSDP): The geometric mean of HSD(time)/CSDP(time) is 16.07 and the confidence interval is (11.93,21.64).

SeDuMi vs CSDP (these results were computed only for problems solved by both SeDuMi and CSDP): The geometric mean of SeDuMi(time)/CSDP(time) is 4.00 and the confidence interval is (3.24,4.95).

problem	SEDUMI	CSDP	HSD	IIP
arch0	0.54	0.15	2.49	2.32
arch2	0.38	0.17	2.47	2.36
arch4	0.40	0.29	2.47	2.36
arch8	0.38	0.17	2.47	2.36
BH2_r14	75.20	78.75	1007.74	747.64
biggs	10.10	1.70	47.95	29.21
bm1	151.72	19.67	671.00	437.20
buck3	9.37	2.03	71.09	47.59
buck4	93.68	12.59	887.16	663.96
buck5	T	135.06	T	T
butcher	M	101.19	M	M
C._1.r14	M	74.44	1240.05	1203.79
C._3.r14	61.50	75.56	777.06	204.75
cancer_100	M	153.81	M	M
checker_1.5	M	419.23	M	M
cnhil10	214.00	23.67	66.01	1.48
cnhil8	9.88	1.52	3.17	0.25
control1	0.01	0.00	0.01	0.01
control10	8.12	8.00	43.30	31.60
control11	13.30	12.27	76.01	49.43
control2	0.02	0.01	0.05	0.04
control3	0.09	0.03	0.20	0.18
control4	0.13	0.09	0.64	0.57
control5	0.33	0.32	1.88	1.45
control6	0.82	0.76	4.29	3.20
control7	1.69	1.58	7.40	6.49
control8	3.12	2.40	16.24	11.50
control9	5.19	1.90	36.51	19.82
copo14	0.57	0.39	11.05	4.68
copo23	50.56	26.52	2178.45	1988.77
cphil10	237.00	25.40	68.42	1.52
cphil12	M	282.00	T	14.04
equalG11	87.50	6.65	448.62	392.65
equalG51	248.00	13.21	844.83	747.99
ex2_1_5	M	81.14	T	T

Table 4.7: Time per iteration

problem	SEDUMI	CSDP	HSD	IIP
filter48_socp	18.19	2.96	94.46	37.22
foot	1470.00	165.20	T	T
G40_mb	M	98.67	14793.66	T
G40mc	930.00	31.68	9255.60	5776.82
gpp100	0.21	0.04	0.74	0.57
gpp124-1	0.27	0.01	1.25	1.02
gpp124-2	0.36	0.08	0.94	1.00
gpp124-3	0.41	0.08	1.00	1.08
gpp124-4	0.35	0.10	1.02	1.09
gpp250-1	3.48	0.41	10.80	7.96
gpp250-2	3.12	0.36	11.30	8.30
gpp250-3	3.47	0.40	9.43	7.89
gpp250-4	3.28	0.42	8.10	7.86
gpp500-1	32.70	3.65	92.32	73.67
gpp500-2	33.30	2.35	94.17	25.03
gpp500-3	32.10	2.44	93.41	51.87
gpp500-4	33.40	2.26	103.56	52.71
H2O_r14g	M	75.86	1519.38	290.09
H2O+_r14	M	76.76	1225.31	281.75
H3O+_r16	253.00	693.75	6190.25	7949.22
hamming_7_5_6	9.73	1.08	35.02	40.18
hamming_9_8	39.86	4.54	366.33	439.15
hand	383.00	27.11	2101.53	2093.20
inc_1200	M	57.38	59617.80	31738.87
inc_600	782.45	8.69	1272.13	1211.54
mater_3	0.50	0.66	2842.04	2420.13
mater_4	2.89	13.63	T	T
mater_5	10.00	114.48	T	T
maxG11	77.70	2.99	T	486.97
maxG32	1090.00	28.65	7924.72	5988.07
maxG51	123.00	8.18	T	684.95
mcp100	0.09	0.03	0.684444444	0.41
mcp124-1	0.16	0.05	0.77	0.57
mcp124-2	0.15	0.06	1.05	0.75
mcp124-3	0.15	0.05	0.98	0.82

Table 4.8: Time per iteration

problem	SEDUMI	CSDP	HSD	IIP
mcp124-4	0.17	0.06	1.79	0.80
mcp250-1	1.19	0.24	6.76	4.43
mcp250-2	1.24	0.24	7.69	5.62
mcp250-3	1.21	0.24	7.19	5.99
mcp250-4	1.22	0.24	7.43	6.02
mcp500-1	15.30	0.92	38.13	32.97
mcp500-2	15.40	1.39	84.62	72.85
mcp500-3	15.30	1.38	91.91	87.84
mcp500-4	15.50	1.38	101.83	77.77
neosfbr20	M	82.80	1443.62	T
neu1	48.30	18.85	270.27	164.92
neu1g	48.60	16.51	301.65	265.65
neu2	48.00	13.88	244.99	181.63
neu2c	63.10	39.00	679.75	140.44
neu2g	52.40	14.31	258.70	205.08
neu3	727.00	165.45	358.28	30.58
neu3g	M	187.69	327.06	10.12
NH2_r14	M	100.20	1008.44	253.10
NH3_1-r16	311.00	937.50	6877.82	5821.56
qap10	2.12	0.56	11.45	15.47
qap5	0.02	0.01	0.10	0.08
qap6	0.06	0.02	0.33	0.30
qap7	0.13	0.06	1.02	1.45
qap8	0.41	0.11	2.06	4.04
qap9	0.86	0.22	4.85	9.53
qpG11	491.00	26.18	389.24	307.75
qpG51	945.00	1290.00	711.81	869.39
rabmo	207.00	31.21	T	T
reimer5	M	13.04	T	T
rendl1_600_0	M	5.27	214.67	131.71
rose13	22.70	3.60	72.30	65.84
rose15	93.30	20.82	261.83	235.79
shmup3	213.00	32.97	659.27	699.73
shmup4	M	83.66	T	8433.34
shmup5	M	0.00	T	M

Table 4.9: Time per iteration

problem	SEDUMI	CSDP	HSD	IIP
ss30	4.26	0.88	12.18	9.05
st_jcbpaf2	63.60	82.80	T	M
swissroll	159.00	23.94	2116.508571	1807.04
taha1a	M	44.88	826.14	942.31
taha1b	M	115.53	0.15	M
theta1	0.03	0.01	2.13	0.20
theta2	0.32	0.09	17.59	3.02
theta3	2.81	0.61	63.93	25.54
theta4	13.70	2.17	177.84	65.25
theta5	46.80	6.18	452.80	184.61
theta6	137.00	13.88	995.94	499.25
thetaG11	86.00	8.04	8414.08	1007.92
thetaG51		85.14	76.66	T
torusg3_15	M	322.78	T	T
torusg3_8	18.57	3.13	73.65	96.53
toruspm3_15_50	M	229.41	T	T
toruspm3_8_50	19.86	2.86	70.87	109.17
trto3	4.05	0.95	31.24	39.49
trto4	53.61	7.93	436.44	447.14
trto5	M	62.17	22897.77	T
truss1	0.01	0.01	0.02	0.02
truss2	0.02	0.01	0.06	0.05
truss3	0.01	0.00	0.02	0.02
truss4	0.01	0.00	0.03	0.03
truss5	0.04	0.04	0.52	0.46
truss6	0.04	0.02	0.40	0.41
truss7	0.03	0.02	0.18	0.20
truss8	0.31	0.28	3.69	3.64
vibra3	10.29	2.91	76.09	62.83
vibra4	115.79	17.68	659.88	803.90
vibra5	T	151.90	43353.68	31268.31
yalsdp	267.00	106.88	772.73	852.33

Table 4.10: Time per iteration

problem	SEDUMI	CSDP	HSD	IIP
arch0	29	27	26	51
arch2	27	25	24	46
arch4	26	25	23	44
arch8	28	25	26	48
BH2_r14	31	40	*	58
biggs	30	66	34	*
bm1	87	61	*	*
buck3	75	61	*	114
buck4	76	49	*	*
buck5	*	*	*	*
butcher	*	59	*	*
C._1.r14	*	36	31	*
C._3.r14	26	36	*	*
cancer_100	*	21	*	*
checker_1.5	*	26	*	*
cnhil10	18	21	18	*
cnhil8	17	19	17	*
control1	26	19	28	25
control10	*	29	*	*
control11	*	26	*	*
control2	30	24	32	33
control3	33	24	*	34
control4	33	24	36	43
control5	*	25	*	*
control6	*	29	*	*
control7	41	26	39	38
control8	*	29	*	37
control9	36	77	*	38
copo14	15	20	12	28
copo23	16	23	15	*
cphil10	6	5	7	12
cphil12	*	5	*	12
equalG11	16	20	*	28
equalG51	42	24	*	28
ex2_1_5	*	44	*	*

Table 4.11: Iteration counts

problem	SEDUMI	CSDP	HSD	IIP
filter48_socp	31	49	35	*
foot	*	25	*	*
G40_mb	*	30	*	*
G40mc	20	19	*	*
gpp100	56	19	29	*
gpp124-1	63	19	*	*
gpp124-2	54	18	*	*
gpp124-3	33	17	*	*
gpp124-4	55	19	31	*
gpp250-1	33	23	*	*
gpp250-2	29	18	*	*
gpp250-3	43	22	*	*
gpp250-4	40	22	*	*
gpp500-1	44	37	*	*
gpp500-2	43	26	*	*
gpp500-3	33	26	*	*
gpp500-4	24	21	*	*
H2O_r14g	*	29	22	*
H2O+_r14	*	34	32	*
H3O+_r16	26	32	*	*
hamming_7_5_6	6	13	8	11
hamming_9_8	7	16	10	10
hand	30	27	*	*
inc_1200	*	*	*	*
inc_600	57	*	*	*
mater_3	27	23	*	*
mater_4	30	27	*	*
mater_5	33	29	*	*
maxG11	13	16	*	26
maxG32	14	17	*	*
maxG51	16	17	*	31
mcp100	13	13	9	19
mcp124-1	13	14	10	22
mcp124-2	13	13	10	21
mcp124-3	13	14	10	20

Table 4.12: Iteration counts

problem	SEDUMI	CSDP	HSD	IIP
mcp124-4	14	14	10	19
mcp250-1	15	15	11	23
mcp250-2	14	14	10	22
mcp250-3	14	14	10	21
mcp250-4	14	14	11	23
mcp500-1	16	16	11	26
mcp500-2	15	16	12	25
mcp500-3	15	15	11	24
mcp500-4	14	15	11	25
neosfbr20	*	25	30	*
neu1	*	*	*	*
neu1g	22	63	21	*
neu2	*	25	20	*
neu2c	*	*	*	*
neu2g	29	36	29	138
neu3	22	*	19	*
neu3g	*	65	19	*
NH2_r14	*	34	29	*
NH3_1-r16	25	32	*	*
qap10	*	16	*	22
qap5	12	13	12	17
qap6	*	17	*	*
qap7	*	17	*	104
qap8	*	16	*	*
qap9	*	17	*	29
qpG11	14	17	13	26
qpG51	22	20	17	*
rabmo	21	28	*	*
reimer5	*	67	*	*
rendl1_600_0	83	22	*	*
rose13	*	52	24	*
rose15	*	49	26	*
shmup3	61	91	55	*
shmup4	*	82	*	*
shmup5	*	*	*	*

Table 4.13: Iteration counts

problem	SEDUMI	CSDP	HSD	IIP
ss30	27	22	*	48
st_jcbpaf2	25	50	*	*
swissroll	*	*	*	*
taha1a	18	43	19	*
taha1b	*	47	11	*
theta1	14	14	13	21
theta2	15	16	12	22
theta3	15	16	13	23
theta4	16	17	13	23
theta5	16	17	14	22
theta6	16	17	14	21
thetaG11	15	23	*	*
thetaG51	*	35	14	*
torusg3_15	*	18.00	*	*
torusg3_8	14	15	14	64
toruspm3_15_50	*	17	*	*
toruspm3_8_50	14	15	11	22
trto3	59	33	*	*
trto4	72	*	*	*
trto5	*	*	*	*
truss1	11	12	8	14
truss2	18	15	14	16
truss3	14	16	12	20
truss4	12	13	8	16
truss5	20	18	17	23
truss6	28	23	22	46
truss7	25	23	19	41
truss8	23	20	19	26
vibra3	69	*	*	*
vibra4	95	*	*	*
vibra5	*	*	*	*
yalsdp	17	16	15	*

Table 4.14: Iteration counts

problem	SEDUMI	CSDP	HSD	IIP
arch0	15.70	4.09	64.74	118.57
arch2	10.30	4.15	59.35	108.52
arch4	10.50	7.34	56.88	103.69
arch8	10.70	4.27	64.27	113.15
BH2_r14	2330.00	3150.00	*	43362.95
biggs	303.00	112.00	1630.25	*
bm1	13200.00	1200.00	*	*
buck3	703.00	124.00	*	5425.3
buck4	7120.00	617.00	*	*
buck5	*	10400.00	*	*
butcher	*	5970.00	*	*
C._1.r14	*	2680.00	38441.52	*
C._3.r14	1600.00	2720.00	*	*
cancer_100	*	3230.00	*	*
checker_1.5	*	10900.00	*	*
cnhil10	3860.00	497.00	1188.10	*
cnhil8	168.00	28.90	53.88	*
control1	0.36	0.04	0.41	0.29
control10	*	232.00	*	*
control11	*	319.00	*	*
control2	0.50	0.12	1.53	1.45
control3	3.02	0.72	*	6.1
control4	4.13	2.15	23.15	24.56
control5	*	7.98	*	*
control6	*	22.00	*	*
control7	69.20	41.10	288.63	246.48
control8	128.00	69.60	*	425.6
control9	187.00	146.00	*	753.04
copo14	8.57	7.85	132.63	131.05
copo23	809.00	610.00	32676.72	*
cphil10	1420.00	127.00	478.97	18.24
cphil12	*	1410.00	*	168.53
equalG11	1400.00	133.00	*	10994.24
equalG51	10400.00	317.00	*	20943.85
ex2_1_5	*	3570.00	*	*

Table 4.15: CPU time, seconds

problem	SEDUMI	CSDP	HSD	IIP
filter48_socp	564.00	145.00	3306.18	*
foot	*	4130.00	*	*
G40_mb	*	2960.00	*	*
G40mc	18600.00	602.00	*	*
gpp100	11.80	0.78	21.35	*
gpp124-1	16.90	4.17	*	*
gpp124-2	19.40	1.38	*	*
gpp124-3	13.60	1.39	*	*
gpp124-4	19.10	1.81	31.73	*
gpp250-1	115.00	9.37	*	*
gpp250-2	90.60	6.52	*	*
gpp250-3	149.00	8.79	*	*
gpp250-4	131.00	9.13	*	*
gpp500-1	1440.00	135.00	*	*
gpp500-2	1430.00	61.00	*	*
gpp500-3	1060.00	63.50	*	*
gpp500-4	802.00	47.50	*	*
H2O_r14g	M	2200.00	33426.33	*
H2O+_r14	M	2610.00	39209.81	*
H3O+_r16	6590.00	22200.00	*	*
hamming_7_5_6	58.40	14.00	280.12	442.03
hamming_9_8	279.00	72.70	3663.30	4391.47
hand	11500.00	732.00	*	*
inc_1200	*	*	*	*
inc_600	4460.00	*	*	*
mater_3	13.60	15.20	*	*
mater_4	86.70	368.00	*	*
mater_5	330.00	3320.00	*	*
maxG11	1010.00	47.80	*	12661.26
maxG32	15200.00	487.00	*	*
maxG51	1970.00	139.00	*	21233.39
mcp100	1.18	0.40	6.16	7.76
mcp124-1	2.02	0.76	7.68	12.43
mcp124-2	2.00	0.73	10.51	15.8
mcp124-3	1.91	0.75	9.81	16.36

Table 4.16: CPU time, seconds

problem	SEDUMI	CSDP	HSD	IIP
mcp124-4	2.31	0.78	17.85	15.23
mcp250-1	17.80	3.56	74.36	101.81
mcp250-2	17.30	3.32	76.94	123.74
mcp250-3	17.00	3.42	71.90	125.69
mcp250-4	17.10	3.34	81.69	138.53
mcp500-1	244.00	14.70	419.43	857.13
mcp500-2	231.00	22.20	1015.40	1821.13
mcp500-3	229.00	20.70	1011.03	2108.25
mcp500-4	217.00	20.70	1120.11	1944.2
neosfbr20	*	2070.00	*	*
neu1	*	1150.00	*	*
neu1g	1070.00	1040.00	6334.56	*
neu2	*	347.00	4899.81	*
neu2c	*	3900.00	*	*
neu2g	1520.00	515.00	7502.42	28300.38
neu3	16000.00	*	6807.29	*
neu3g	*	12200.00	6214.10	*
NH2_r14	*	3410.00	29244.74	*
NH3_1-r16	7770.00	30000.00	*	*
qap10	*	8.98	*	340.28
qap5	0.26	0.08	1.24	1.41
qap6	*	0.29	*	*
qap7	*	1.05	*	151.03
qap8	*	1.79	*	*
qap9	*	3.67	*	276.3
qpG11	6870.00	445.00	5060.17	8001.57
qpG51	20800.00	25800.00	12100.82	*
rabmo	*	874.00	*	*
reimer5	*	874.00	*	*
rendl1_600_0	3630.00	116.00	*	*
rose13	*	187.00	1735.27	*
rose15	*	1020.00	6807.55	*
shmup3	13000.00	3000.00	36260.03	*
shmup4	*	6860.00	*	*
shmup5	*	*	*	*

Table 4.17: CPU time, seconds

problem	SEDUMI	CSDP	HSD	IIP
ss30	115.00	19.40	*	434.17
st_jcbpaf2	1590.00	4140.00	*	*
swissroll	*	*	*	*
taha1a	1090.00	1930.00	15696.65	*
taha1b	*	5430.00	1.67	*
theta1	0.41	0.18	27.71	4.29
theta2	4.82	1.36	211.06	66.35
theta3	42.10	9.82	831.07	587.38
theta4	219.00	36.90	2311.96	1500.78
theta5	749.00	105.00	6339.16	4061.39
theta6	2190.00	236.00	13943.17	10484.34
thetaG11	1290.00	185.00	*	*
thetaG51	*	2980.00	1073.22	*
torusg3_15	*	5810.00	*	*
torusg3_8	260.00	47.00	1031.11	6177.97
toruspm3_15_50	*	3900.00	*	*
toruspm3_8_50	278.00	42.90	779.56	2401.75
trto3	239.00	31.30	*	*
trto4	3860.00	*	*	*
trto5	*	*	*	*
truss1	0.10	0.08	0.19	0.21
truss2	0.30	0.10	0.80	0.77
truss3	0.14	0.02	0.29	0.42
truss4	0.10	0.03	0.20	0.51
truss5	0.83	0.66	8.87	10.62
truss6	1.22	0.36	8.73	18.73
truss7	0.86	0.39	3.45	8.31
truss8	7.04	5.66	70.10	94.73
vibra3	710.00	*	*	*
vibra4	11000.00	*	*	*
vibra5	*	*	*	*
yalsdp	4539.00	1710.00	11590.97	*

Table 4.18: CPU time, seconds

Obviously, CSDP is much faster than all the others.

CHAPTER 5

CONCLUSIONS

In terms of robustness, HSD and IIP are comparable; in terms of efficiency, HSD is faster than IIP. Generally, HSD takes more time than IIP in each iteration, but HSD uses fewer iterations than IIP. This leads to an overall decrease in CPU time for HSD compared to IIP.

CSDP is significantly faster than all the other three methods, so we can see the computer language and implementation details play a great role in the computation performance.

Bibliography

[1] Brian Borchers. CSDP, A C library for semidefinite programming. *Optimization Methods and Software*, 11:613–623, 1999.

[2] Brian Borchers. SDPLIB 1.2 a library of semidefinite programming test problems. *Optimization Methods and Software*, 11-2:683–690, 1999. http://infohost.nmt.edu/∼sdplib.

[3] Brian Borchers and Joseph Young. Implementation of a primal-dual method for SDP on a parallel architecture. 2005.

[4] N. Brixius, Florian A. Potra, and Rongqin Sheng. SDPHA a MATLAB implementation of homogeneous interior-point algorithms for semidefinite programming. *Optimization Methods and Software*, 11-2:583–596, 1999.

[5] Vasek Chvatal. *Linear Programming*. W.H. Freeman, New York, 1983.

[6] Etienne de Klerk. *Aspects of semidifinite programming : interior point algorithms and selected applications*, volume 65 of *Applied Optimization*. Kluwer Academic Publishers, 2002.

[7] Etienne de Klerk, C. Roos, and T. Terlaky. Initialization in semidefinite programming via a self-dual skew-symmetric embedding. *Operations Research Letters*, 20:213–221, 1997.

[8] D. Goldfarb and K. Scheinberg. Interior point trajectories in semidefinite programming. *SIAM Journal on Optimization*, 8:871–886, 1998.

[9] C. Helmberg, F. Rendl, R.J. Vanderbei, and H. Wolkowicz. An interior point method for semidefinite programming. *SIAM Journal on Optimization*, 6:342–361, 1996.

[10] M. Kojima, S. Shindoh, and S. Hara. Interior-point methods for the monotone semidefinite linear complementarity problem in symmetric matrices. *SIAM Journal on Optimization*, 7:86–125, 1997.

[11] M.Halicka, E. de Klerk, and C. Roos. On the convergence of the central path in semidefinite optimization. *SIAM Journal on Optimization*, 12:1090–1099, 2002.

[12] Hans D. Mitellmann. Several sdp-codes on sparse and other sdp problems. ftp://plato.asu.edu/pub.

[13] H. D. Mittelmann. An independent benchmarking of SDP and SOCP solvers. *Mathematical Programming*, 95:407–430, 2003.

[14] R.D.C. Monteiro. Primal-dual path-following algorithms for semidefinite programming. *SIAM Journal on Optimization*, 7:663–678, 1997.

[15] Stephen G. Nash and Ariela Softer. *Linear and Nonlinear Programming.* McGraw-Hill, New York, 1996.

[16] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer Verlag, New York, 1999.

[17] Florian A. Potra and Rongqin Sheng. On homogeneous interior-point algorithms for semidefinite programming. *Optimization Methods and Software*, 9:161–184, 1998.

[18] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11:625–653, 1999.

[19] K. C. Toh, M. J. Todd, and R. H. Tutuncu. SDPT3- A MATLAB software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11:545–581, 1999.

[20] Lieven Vandenberghe and Stephen Boyd. Semidefinite programmming. *SIAM Review*, 38:49 – 95, 1996.

[21] Henry Wolkowicz, Romesh Saigal, and Lieven Vandenberghe, editors. *Handbook of Semidefinite Programming*. Kluwer Academic Publishers, 2000.

[22] Stephen J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia, 1997.

[23] Y. Ye and M.Todd. An O($\sqrt{nl}$)-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 19:53–67, 1994.