Master's Thesis

A hybrid algorithm of gradient and Newton methods for semidefinite programs

Guidance

Professor Nobuo YAMASHITA

Kazuma IWASAKI

Department of Applied Mathematics and Physics

Graduate School of Informatics

Kyoto University

February 2015

Abstract

Semidefinite programs (SDP) include many type problems, such as linear programming problems and second-order cone programming problems. Moreover, it has a lot of applications, e.g., LMI-constrained problems and minimal (maximal) eigenvalue problems. The most popular method for solving SDP is the interior point method. However, it cannot get a high accurate solution to some SDPs since it suffers from a large amount of numerical erorrs.

In this paper, we propose a novel Newton type method that gets more accurate solution to SDP than that obtained by the interior point method. For this purpose, we focus on the nonlinear programming problem equivalent to the SDP, which has been presented by Burer and Monteiro. Then, we adopt a sequential quadratic programming method (SQP method) for the equivalent problem. We show that the SQP subproblem can be solved in $\mathcal{O}(n^3 + m^3 + nm \max\{nm^{1/2}, m^{3/2}\})$ by exploiting special structure of the subproblem, where n is a size of variable matrix and m is a number of equalities constraints in the original SDP. We also adopt the interior point method or the first-order method for the original SDP to get an initial point of the SQP. This is because the equivalent nonlinear programming problem is not convex, and hence the SQP may need many iterations to get a point near a solution to the problem. Finally, we present numerical results that show the validity of the proposed method.

Contents

1 Introduction

In this paper, we consider the following semidefinite programming (SDP):

$$\begin{array}{ll} \min & C \bullet X, \\ \text{subject to} & A_i \bullet X = b_i, i = 1 \dots m, X \succeq 0, \end{array} \tag{1}$$

where $C, \{A_i\}_{i=1}^m$ are $n \times n$ symmetric matrices, b is m-dimensional vector, the operator $\bullet$ denotes the inner product of matrices, i.e. $C \bullet D = \mathrm{trace}(C^T D)$, and $X \succeq 0$ indicates that X is symmetric and positive semidefinite. The SDP has a lot of applications such as LMI-constrained problems [3], material optimizations [7], structual optimizations [4], and so on. Moreover, the SDP includes many types of problems, for example, the second order cone problems (SOCP) and the linear programming problems (LP).

Until now, many algorithms for solving SDP have been proposed. For example, the interior point method (IPM), the gradient method, the Newton method are well known. Among them, the most popular method is IPM. There exists several software packages based on IPM, for example SeDuMi [8], SDPT3 [9] and SDPA [10]. The IPM can obtain a solution after dozens of iterations. Actually, even if a number of variables is several thousand, the IPM can obtain solution in practical time. However, the IPM is hard to obtain accurate solution to some SDPs, and cannot choose an initial point arbitrarily. Furthermore, if the number of variables is more than tens of thousands, then each iteration of IPM takes a long time since IPM is a Newton type method.

A first-order method, for example boundary point method [6] and block-decomposition algorithm [5], is suitable for solving large scale SDPs. However these algorithms cannot obtain an accurate solution.

It is well-known that the SDP can be transformed into the equality constrained nonlinear programming. Note that a symmetric positive definite matrix can be represented as $X = VV^T$, where V is an $n \times n$ matrix. Then, SDP is equivalent to the following nonlinear programming:

$$\begin{array}{ll} \text{minimize} & C \bullet VV^T, \\ \text{subject to} & A_i \bullet VV^T = b_i, i = 1 \dots m. \end{array} \tag{2}$$

Since the constraints of problem (2) are equality constraints only, the KKT conditions for problem (2) become nonlinear equations. Furthermore, we can replace V with $R \in \mathbb{R}^{n \times r}$ where $n > r > \sqrt{2m}$.

$$\begin{array}{ll} \text{minimize} & C \bullet RR^T, \\ \text{subject to} & A_i \bullet RR^T = b_i, i = 1 \dots m. \end{array} \tag{3}$$

If $n \gg r$, the number of variables is reduced. This is called low-rank model [1], and is effective when m is small. Burer and Monteiro [2] introduced an augmented Lagrangian method for solving this model, and solve its subproblem by the gradient method. Their algorithm is effective for large scale problems, but is difficult to obtain an accurate solution.

In this paper, we propose an algorithm based on a Newton-type method for solving (3). We expect that the proposed algorithm obtains a more accurate solution and convergences fast. However, problem (3) is nonconvex, so the algorithm might require many number of iterations to convergence near the optimal solution. It takes much time solving Newton equation many times. Therefore we adopt the gradient method or the interior point method to approach near to the optimal solution. Note that we apply neither the gradient method nor the interior point method to nonconvex problem (2), since the number of iteration may not be reduced because of nonconvexity. We apply the first-order method [5] and the interior point method to the original convex SDP(1). And if an iteration point is near solution, we change the Newton type method to problem (2).

This paper is organized as follows. In Section 2, we introduce the existing methods for SDP, low-rank method and first-order method. In Section 3, we first propose a sequential quadratic programming method for low-rank model (3), then propose a hybrid algorithm of gradient and Newton methods. In section 4, we give some numerical results. In Section 5, we conclude the paper.

The notation used in this paper is as follows. The norm of a linear operator $\mathcal{A}$ is defined as

$$\|A\| := \sup_{\|x\| \leq 1} \|Ax\|.$$

The projection operator $\Pi_{\mathcal{S}^n}(X)$ is defined as

$$\Pi_{\mathcal{S}^n}(X) := \arg\min_{\tilde{X} \in \mathcal{S}^n} \{X - \tilde{X}\}.$$

2 Preliminaries

In this section, we give some preliminaries. First, we introduce a dual problem of SDP (1). Second, we introduce the low-rank model and its optimality conditions. Finally, we introduce the block-decomposition algorithm that is an example of the first-order method for SDP.

2.1 The dual problem of SDP

The dual problem of SDP (1) is defined as

$$\begin{aligned} &\text{maximize} \quad b^T y, \\ &\text{subject to} \quad S = C - \sum_{i=1}^{m} y_i A_i, \ S \succeq 0, \end{aligned} \tag{4}$$

where $(S, y) \in \mathcal{S}^n \times \mathbb{R}^m$ are variables. We assume that (1) and (4) have nonempty optimal solution sets with zero duality gap, that is, feasible solutions X^* and (S^*, y^*) such that $C \bullet X^* = b^T y^*$ exist.

2.2 Low-rank model

In this section, we introduce reformulation of SDP to the equality constrained nonlinear programming problem.

2.2.1 Low-rank model equivalent to SDP

First, we think about the rank of extreme points for SDP, using the following definition: for any positive integer l,

$$r_l := \max\{r \text{ integer} : r(r+1)/2 \leq l\}.$$

Theorem 1. *[1, Theorem 1] If $\bar{X}$ is an extreme point of SDP, then rank($\bar{X}$) $\leq r_m$*

From this theorem, since the optimal value of SDP is attained at the extreme point, the following low-rank semidefinite programming problem is equivalent to SDP for $r \geq r_m$:

$$\begin{array}{rll} & \text{minimize} & C \bullet X, \\ (\text{LRSDP}_r) & \text{subject to} & A_i \bullet X = b_i, i = 1, \ldots, m, \\ & & X \succeq 0, \text{rank}(X) \leq r. \end{array} \tag{5}$$

Since the constraint $X \succeq 0$ is difficult to handle directly, we replace $X \succeq 0$ with the condition that $X = VV^T$ for some $V \in \mathbb{R}^{n \times n}$. Then, we reformulate LRSDP_r as the following nonlinear programming problem:

$$\begin{array}{ll} \text{minimize} & C \bullet (VV^T), \\ \text{subject to} & A_i \bullet (VV^T) = b_i, i = 1, \ldots, m. \end{array} \tag{6}$$

Furthermore, we can choose V having its last $n - r$ columns equal to zero, because $\text{rank}(X) \leq r$. Therefore, we reformulate (6) with using $R \in \mathbb{R}^{n \times r}$ as the nonlinear programming problem:

$$\begin{array}{rll} & \text{minimize} & C \bullet (RR^T), \\ (N_r) & \text{subject to} & A_i \bullet (RR^T) = b_i, i = 1, \ldots, m, \end{array}$$

where the decision variable R is an $n \times r$ matrix. The advantage of (N_r) over (6) is to reduce the number of variables. Actually, the number of variables in (6) is n^2, however that in (N_r) is nr.

2.2.2 Optimality conditions

We present the optimality conditions of the nonlinear programming problem (N_r) for a fixed r. First, we show that any extra local minima is not introduced by changing variables $X = RR^T$.

Proposition 1. *[2, Proposition 2.3] Suppose $\bar{X} = \bar{R}\bar{R}^T$, where $\bar{X}$ is feasible for ($LRSDP_r$) and $\bar{R}$ is feasible for (N_r). Then $\bar{X}$ is a local minimum of ($LRSDP_r$) if and only if $\bar{R}$ is a local minimum of (N_r).*

Next, we introduce how large must we take r so that the local minima of (LRSDP_r) are guaranteed to be global minima of SDP.

Proposition 2. *[2, Proposition 3.3] Let $\bar{X}$ be an extreme point of SDP and assume that $\{X^k\}$ is a sequence of feasible points for SDP converging to $\bar{X}$. Then, there exists a sequence $\{Y^k\}$ of feasible points for SDP converging to $\bar{X}$ and satisfying $C \bullet Y^k = C \bullet X^k$ and* $\mathrm{rank}(Y^k) \leq r_{m+1}$ for all k.

Then we can now introduce the relationship between the local minima of (LRSDP_r) and the optimal solution of original SDP, if r is sufficiently large, by using proposition 2.

Theorem 2. *[2, Theorem 3.4] Suppose $\bar{X}$ is a local minimum of ($LRSDP_r$) for some $r \geq r_{m+1}$. Then, $\bar{X}$ is contained in the relative interior of a face $\bar{F}$ of SDP over which the objective function is constant. Moreover, if the dimension of $\bar{F}$ is zero then $\bar{X}$ is an optimal extreme point of SDP.*

This theorem states that if r is large enough, then the local optimal solution of (N_r) becomes the optimal solution of SDP. However, it does not say that the KKT point of (N_r) becomes the optimal solution of SDP. Since standard algorithms for the nonlinear programming problem find a KKT point, the low-rank model may not provide an optimal solution of SDP if the initial point of the algorithm is not chosen appropriately.

2.3 First-order method

In this section, we introduce the first-order method. The first-order method for solving SDP has been presented, e.g., the boundary point method [6] and the block-decomposition algorithm [5]. Now, we introduce the block-decomposition algorithm, since it is better than any other first-order method in the terms of numerical results.

First, we define $\mathcal{A} : \mathcal{S}^n \to \mathbb{R}^m$ as follows,

$$\mathcal{A} = \begin{pmatrix} A_1 \\ \vdots \\ A_m \end{pmatrix}.$$

Then, we reformulate SDP(1) to

$$\begin{array}{ll} \text{minimize} & C \bullet X, \\ \text{subject to} & \mathcal{A}X = b, X \succeq 0. \end{array} \tag{7}$$

Algorithm 1. *The block-decomposition algorithm [5]*

(Step0) Let $X_0 \in \mathcal{S}^n, y_0 \in \mathbb{R}^m, 0 < \sigma < 1$ *be given, and set* $k = 1$ *and*

$$\tilde{\lambda} = \frac{\sigma}{\|\mathcal{A}\|}.$$

(Step1) Compute $\tilde{y}_k, \tilde{x}_k$ *by*

$$\tilde{y}_k = y_{k-1} - \tilde{\lambda}(\mathcal{A}X_k - b),$$
$$\tilde{X}_k = \Pi_{\mathcal{S}^n}[X_{k-1} - \tilde{\lambda}(C - \sum_{\imath=1}^{m}(\tilde{y}_k)_i A_i)].$$

(Step2)Choose λ_k *to be the largest* $\lambda > 0$ *such that*

$$\sigma^2(\|\tilde{X}_k + X_{k-1}\|^2 + \|\tilde{y}_k + y_{k-1}\|^2) \geq$$
$$\left\|\frac{\lambda}{\tilde{\lambda}}(X_{k-1} - \tilde{X}_k) + \tilde{X}_k - X_{k-1}\right\|^2 + \|\lambda(\mathcal{A}\tilde{X}_k - b) + \tilde{y}_k - y_{k-1}\|^2.$$

(Step3) Update X_k, y_k *such that*

$$X_k = X_{k-1} - \frac{\lambda_k}{\tilde{\lambda}}(X_{k-1} - \tilde{X}_k),$$
$$y_k = y_{k-1} - \lambda_k(\mathcal{A}\tilde{X}_k - b).$$

(step4) Set $k = k + 1$ *and go to (Step1).*

The global convergence of this algorithm is proved in $[5, Theorem 3.3]$. From the computational point of view, we define the norm of y as $\|y\| = \sqrt{y^T\mathcal{U}y}$, where $\mathcal{U} : \mathbb{R}^m \to \mathbb{R}^m$, and update $\mathcal{U}$ at each iterations.

3 Algorithm

In this section, we first propose the sequential quadratic programming method (SQP method) for the low-rank model (N_r). Then, we propose a hybrid algorithm of the gradient method / the interior point method and SQP method.

3.1 SQP method for low-rank model

Low-rank model (N_r) is the optimization problem whose variable is a matrix $R \in \mathbb{R}^{n\times r}$. First, we transform (N_r) to a problem whose variable is a vector $\gamma \in \mathbb{R}^{nr}$ to explain SQP method easily.

Let $r^i, i = 1, \ldots, r$ be an n-dimensional vectors such that $R = [r^1 r^2 \ldots r^r]$. Then, for any $Z \in R^{n \times n}$,

$$\begin{aligned} Z \bullet (RR^T) &= \mathrm{tr}(ZRR^T) \\ &= \mathrm{tr}(R^T ZR) \\ &= \mathrm{tr}\left(\begin{bmatrix} (r^1)^T \\ \vdots \\ (r^r)^T \end{bmatrix} [Zr^1 \ldots Zr^r] \right) \\ &= \sum_{i=1}^{r} (r^i)^T Z r^i. \end{aligned}$$

It then follows that, (N_r) is transformed as follows:

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{r} (r^i)^T C r^i, \\ \text{subject to} & \displaystyle\sum_{j=1}^{r} (r^j)^T A_i r^j = b_i, i = 1, \ldots, m. \end{array} \tag{8}$$

Let $\gamma \in \mathbb{R}^{nr}$ and let matrices $H \in \mathbb{R}^{nr \times nr}, G_i \in \mathbb{R}^{nr \times nr} (i = 1, \ldots m)$ be defined as follows:

$$\gamma = \begin{pmatrix} r^1 \\ \vdots \\ r^r \end{pmatrix},$$

$$H = \begin{pmatrix} C & & 0 \\ & \ddots & \\ 0 & & C \end{pmatrix},$$

$$G_i = \begin{pmatrix} A_i & & 0 \\ & \ddots & \\ 0 & & A_i \end{pmatrix}.$$

Then, (8) is rewritten as

$$\begin{array}{ll} \text{minimize} & \gamma^T H \gamma \\ \text{subject to} & \gamma^T G_i \gamma = b_i, i = 1, \ldots, m. \end{array} \tag{9}$$

The KKT conditions for this problem are as follows:

$$H\gamma - \sum_{i=1}^{m} y_i G_i \gamma = 0,$$
$$\gamma^T G_i \gamma - b_i = 0, i = 1, \dots m,$$

where $y_i, i = 1, \dots m$ are Lagrangian multipliers corresponding to the equality constraints of (9).

We solve the problem (9) by the standard SQP method. The SQP method solves the quadratic programming that is quadratic approximation of original problem (9) at each iteration, and adopts its solution as a search direction. The quadratic programming subproblem of the SQP method is written as

$$\text{(QP)} \quad \begin{array}{ll} \text{minimize} & \frac{1}{2}\Delta\gamma_k M \Delta\gamma_k + \gamma_k^T H \Delta\gamma_k, \\ \text{subject to} & \gamma_k^T G_i \gamma_k - b_i + \gamma_k^T G_i \Delta\gamma_k = 0, i = 1, \dots, m, \end{array}$$

where M is the Hessian of the Lagrangian function with respect to γ or its approximation. Note that the Lagrangian function is

$$L(\gamma_k, y) = \gamma_k^T H \gamma_k - \sum_{i=1}^{m} y_i (\gamma_k^T G_i \gamma_k - b_i),$$

and its Hessian is given as

$$\nabla^2_{\gamma_k} L(\gamma_k, y) = H - \sum_{i=1}^{m} y_{ik} G_i.$$

The KKT condition for (QP) is

$$\begin{cases} M\Delta\gamma_k + H\gamma_k + \sum_{i=1}^{m} y_i G_i \gamma_k = 0, \\ \gamma_k^T G_i \gamma_k - b_i + \gamma_k G_i \Delta\gamma_k = 0, i = 1, \dots, m. \end{cases} \tag{10}$$

Let $\bar{\Delta\gamma}_k$ be the solution of (QP). Then we get a next point γ_{k+1} by

$$\gamma_{k+1} = \gamma_k + t_k \bar{\Delta\gamma}_k,$$

where t_k is a stepsize. For a global convergence, we use a stepsize t_k that satisfies the Armijo rule. Let $P(\gamma; \rho)$ be the l_1 penalty function defined by

$$P(\gamma; \rho) = \gamma^T H \gamma + \rho \left(\sum_{i=1}^{m} |\gamma^T G_i \gamma - b_i| \right).$$

Moreover, we define $P(\gamma, \Delta\gamma)$ by using P_l which is a linear approximation of the penalty function, that is

$$P_l(\gamma, \Delta\gamma; \rho) = \gamma^T H\gamma + \gamma^T H\Delta\gamma + \rho\left(\sum_{i=1}^{m} |\gamma^T G_i\gamma - b_i + \gamma^T G_i\Delta\gamma|\right),$$
$$\Delta P(\gamma, \Delta\gamma) = P_l(\gamma, \Delta\gamma; \rho) - P(\gamma; \rho).$$

Then, Armijo rule with $\xi \in (0, 1)$ is written as

$$P(\gamma_k + \alpha\Delta\gamma_k; \rho) \leq P(\gamma_k; \rho) - \xi t_k \Delta P(\gamma_k, \Delta\gamma_k). \tag{11}$$

It is well-known that if $\rho > \|y\|$, then

$$\Delta P(\gamma_k, \Delta\gamma_k) \leq -\gamma_k^T H\gamma_k.$$

Hence, we can get the stepsize that satisfies Armijo rule, if we choose ρ that satisfies $\rho > \|y\|$. Furthermore, if $\Delta\gamma_k = 0$, γ_k becomes a stationary point of the original problem (9) due to (10). Thus we may expect γ_k is a good approximation of the optimal solution if $\|\Delta\gamma\|$ is small enough.

The details of the SQP method for low-rank model (9) are as follows.

Algorithm 2. *Algorithm of the SQP method for low-rank model (9)*

(Step0) Set k=0. Choose an initial point $X_0 \in \mathcal{S}^n, y_0 \in \mathbb{R}^m$. Given $\rho > 0, \xi \in (0, 1)$, threshold $\epsilon > 0$. Then calculate γ_0 from $X_0 = R_0 R_0^T$.

(Step1) Solve the subproblem (QP) to get a search direction $\Delta\gamma_k$ and Lagrangian multiplier y_{k+1}.

(Step2) If $\|\Delta\gamma\| \leq \epsilon$, stop.

(Step3) Get a stepsize α_k that satisfies Armijo rule (11).

(Step4) Set $\gamma_{k+1} = \gamma_k + \alpha\Delta\gamma_k$.

(Step5) Set $k = k + 1$, then go to (Step1).

If there exists $c_2 > c_1 > 0$, such that for all v

$$c_1\|v\|^2 \leq v^T M_k v \leq c_2\|v\|^2$$

global convergence of the algorithm is guaranteed. On the other hand, if

$$\lim_{k\to\infty} \frac{\|(M_k - \nabla_\gamma^2 L(\gamma_k, y))\Delta\gamma_k\|}{\|\Delta\gamma_k\|} = 0$$

hold, superlinear convergence is guaranteed. For this purpose, we choose M_k as a positive definite matrix that is a good approximation of $\nabla^2_\gamma L(\gamma_k, y)$. Hessian of Lagrange function is

$$\begin{aligned}\nabla^2_r L(r,y) &= H - \sum_{i=1}^{m}(y_k)_i G_i \\ &= \begin{pmatrix} C - \sum_{i=1}^{m}(y_k)_i A_i & & 0 \\ & \ddots & \\ 0 & & C - \sum_{i=1}^{m}(y_k)_i A_i \end{pmatrix},\end{aligned}$$

whitch is not necessarily positive definite. Therefore, we cannot directly use Hessian as M_k. We adopt M_k as follows:

$$M_k = H - \sum_{i=1}^{m} y_{ik} G_i + (\max\{0, -\sigma\} + Error)I, \tag{12}$$

where σ is minimal eigenvalue of $\nabla^2_r L(r,y)$ and $Error = \frac{\sum_{i=1}^{m}\|A_i X - b_i\|}{1+\|b\|} + \|C \bullet X - b^T y\|$. Generally speaking, Hessian of the nonlinear programming problem may not be semidefinite. Unless σ is large, M_k is not positive definite. It seems difficult to get the minimal eigenvalue of $\nabla^2_r L(r,y)$ because $\nabla^2_r L(r,y)$ is $nr \times nr$ matrix. However, we can get the minimal eigenvalue easily by using the special structure of (QP). $\nabla^2_r L(r,y)$ is a block diagonal matrix with all block are $C - \sum_{i=1}^{m}(y_k)_i G_i$. Exploiting this special structure, we can choose σ that make $C - \sum_{i=1}^{m}(y_k)_i A_i + \sigma I$ positive definite. On the other hand, we expect $C - \sum_{i=1}^{m}(y_k)_i G_i$ be semidefinite enough near the solution, because we can regard y_k as the variable of dual problem (4). If γ_k is near the solution, M_k defined by (12) is a good approximation of Hessian with small σ.

Meanwhile, the SQP method takes a large amount of calculation time for solving (QP). The KKT conditions of (QP) are

$$\begin{cases} M_k \Delta\gamma_k + H\gamma_k + \sum_{i=1}^{m}(y_{k+1})_i G_i \gamma_k = 0, \\ \gamma_k^T G_i \gamma_k - b_i + \gamma_k^T G_i \Delta\gamma_k = 0, i = 1, \ldots, m. \end{cases}$$

We transform the KKT conditions with

$$\begin{aligned} N_k &= (G_1\gamma_k \ldots G_m\gamma_k), \\ z &= \begin{pmatrix} b_1 - \gamma_K^T G_1 \gamma_k \\ \vdots \\ b_m - \gamma_K^T G_m \gamma_k \end{pmatrix}. \end{aligned}$$

Then, we get

$$\begin{cases} M_k \Delta\gamma_k + H\gamma_k + N_k y_{k+1} = 0, \\ N_K^T \Delta\gamma_k = z. \end{cases} \tag{13}$$

The solution of (13) is as follows:

$$\begin{aligned} \Delta\gamma_k &= M_k^{-1}(N_k(y_{k+1}) - H\gamma_k), \\ y_{k+1} &= -(N_k^T M_k^{-1} N_k)^{-1}(N_k^T M_k^{-1} H\gamma_k - z). \end{aligned}$$

Now, we focus on the complexity of the calculations. The main computational complexities are as follow:

1. compute N_k,
2. compute M_k^{-1},
3. compute $M_k^{-1}N_k$ and $N_k^T M_k^{-1} N_k$,
4. solve $(N_k^T M_k^{-1} N_k)u = v$ for given v.

First, the computational complexity of computing N_k is $\mathcal{O}(n^2mr)$ if A_i is dense. If A_i is sparse, the computational complexity of computing N_k is $\mathcal{O}(r\sum_{i=1}^m nnz(A_i))$ where $nnz(A_i)$ is the number of non-zero elements of A_i. Second, the computational complexity of computing M_k^{-1} is $\mathcal{O}(n^3)$ because M_k is a block diagonal matrix with all blocks are $C - \sum_{i=1}^m (y_k)_i A_i + \sigma I$. In this computation, we can reduce the computational complexity if $C - \sum_{i=1}^m (y_k)_i A_i$ is sparse. Third, the computational complexity of computing $M_k^{-1}N_k$ is $\mathcal{O}(n^2rm)$. Furthermore the computational complexity of computing $N_k^T M_k^{-1} N_k$ is $\mathcal{O}(m^2nr)$ because the size of N_k and $M_k^{-1}N_k$ is $nr \times m$. Finally, the computational complexity of solving $(N_k^T M_k^{-1} N_k)u = v$ is $\mathcal{O}(m^3)$ because the size of $N_k^T M_k^{-1} N_k$ is $m \times m$.

Collectively, the computational complexity of the all computations for (QP) are $\mathcal{O}(n^3 + n^2mr + m^2nr + m^3)$. If m is small, then r is small. So the computational complexity becomes $\mathcal{O}(n^3)$. The worst case of computational complexity is when $m = \mathcal{O}(n^2)$, then it becomes $\mathcal{O}(n^6)$.

3.2 A hybrid algorithm of gradient and Newton method

In this section, we propose a hybrid algorithm of gradient and Newton method.

Algorithm 2 is the SQP method, for the local fast convergence, we set an initial point near the solution, and take a stepsize α_k near 1. However, it takes a long time that we set an initial point near the solution by gradient method or interior point method, because many iterations required. Furthermore, it takes a long time if an initial point of algorithm 2 is not near the solution, because we solve Newton equation many times.

Algorithm 3. *A hybrid algorithm of gradient and Newton methods*

(Step0) Choose an initial point $X_0 \in \mathcal{S}^n, y_0 \in \mathbb{R}^m$. Given the switching criteria and the stoppint criteria.

(Step1) Until the switching criteria satisfies, do a gradient method or an interior point method for SDP (1).

(Step2) Set the X, y obtained by (Step1) as an initial point, then do Algorithm 2 until the stopping criteria satisfies.

This algorithm is a hybrid algorithm that guarantees global convergence, so we can expect global convergence of this algorithm. The switching criteria and the stopping criteria are important in this algorithm.

4 Numerical experiments

In this section, we implement Algorithm 3 and report some numerical results. The program is coded in MATLAB R2012b and run on a machine with an Intel(R) Core(TM)i7 1.60GHz CPU and 4GB RAM.

We compare the algorithm that we propose and SeDuMi [8] which is based on interior point method, in terms of accuracy of solution. The measure of an accuracy of solution is as follows:

$$Error = \frac{\sum_{i=1}^{m} \|A_i \bullet X - b_i\|}{1 + \|b\|} + \|C \bullet X - b^T y\| \tag{14}$$

In $Error$, $\|A_i \bullet X - b_i\|$ means how the solution invade the equality constraints of primal problem, $\|C \bullet X - b^T y\|$ means a duality gap.

We use two algorithms to show the validity of the proposed algorithm. One is SeDuMi, and the other is Algorithm 3 that uses SeDuMi in Step1. We run SeDuMi until it stops by numerical error. The solution obtained by the SeDuMi is called the most accurate solution of SeDuMi. Algorithm 3 adopt the most accurate solution of SeDuMi as its initial point, and conduct 10 iterations of Algorithm 2.

In this experiment, we use random SDPs [9] as the problems, set the initial point X_0, y_0 also random. The size of variables and equality constraints are in the table 1.

Table 1: The size of variables and equality constraints in test problem

	n	m
Problem 1	100	10
Problem 2	200	10
Problem 3	200	50

We solve fifty randomly generated SDPs, for each size of Problem.
The obtained results are shown in Table 2, Table 3 and Table 4, where

average:	the average of $Error$s defined in (14);
min:	the minimum value of $Error$s;
max:	the maximum value of $Error$s;
stdev:	the standard deviation of $Error$s;
time(sec):	the average of computational time in seconds.

From the tables, we can observe that our proposed algorithm can find a more accurate solution than that of SeDuMi. However, our proposed algrithm takes more time than that of SeDuMi. We conclude that our proposed algorithm is more stable than SeDuMi in point of view of accuracy, but takes more time.

Table 2: Numerical results of Problem1

	SeDuMi	SeDuMi + Algorithm 2
average	1.867×10^{-07}	1.521×10^{-09}
min	1.142×10^{-09}	9.112×10^{-13}
max	3.887×10^{-06}	5.561×10^{-08}
stdev	5.597×10^{-07}	8.023×10^{-09}
time(sec)	3.224	4.144

Table 3: Numerical results of Problem2

	SeDuMi	SeDuMi + Algorithm 2
average	2.833×10^{-07}	5.097×10^{-09}
min	1.092×10^{-08}	7.426×10^{-15}
max	1.078×10^{-06}	1.178×10^{-07}
stdev	2.733×10^{-07}	2.101×10^{-08}
time(sec)	8.043	11.600

Table 4: Numerical results of Problem3

	SeDuMi	SeDuMi + Algorithm 2
average	1.283×10^{-06}	8.721×10^{-09}
min	4.425×10^{-09}	7.283×10^{-12}
max	5.714×10^{-06}	2.191×10^{-07}
stdev	1.252×10^{-06}	3.667×10^{-08}
time(sec)	36.100	47.821

5 Conclusion

In this paper, we proposed the hybrid algorithm of gradient and Newton method for solving the nonlinear programming problem equivalent to SDP. From the numerical results, we see the validity of the proposed method. For a future work, we may consider about the switching criteria which plays an important role in the hybrid algorithm.

Acknowledgements

First of all, I would like to express sincere appreciation to Professor Nobuo Yamashita. Although I sometimes troubled him due to my greenness, he always kindly looked after me and gave me plenty of precise advice. It is an honor to have studied under him. I am grateful to Assistant Professor Ellen Hidemi Fukuda. She was so kind that she gave me plenty of precise advice. Finally, I would like to thank all members of Yamashita Laboratory, my friends and my family for their encouraging words.

References

[1] S. Burer and R. D. C. Monteiro, A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization, Mathematical Programming, 95.2, pp. 329–357, 2003.

[2] S. Burer and R. D. C. Monteiro, Local minima and convergence in low-rank semidefinite programming, Mathematical Programming, 103.3, pp. 427–444, 2005.

[3] B. Fares, A. Pierre and N. Dominikus, An augmented Lagrangian method for a class of LMI-constrained problems in robust control theory, International Journal of Control, 74.4, pp. 348–360, 2001.

[4] Y. Kanno, I. Takewaki, Sequential semidefinite program for maximum robustness design of structures under load uncertainty, Journal of Optimization Theory and Applications, 130.2, pp. 265–287, 2006.

[5] R. D. C. Monteiro, C. Ortiz and B. F. Svaiter, Implementation of a block-decomposition algorithm for solving large-scale conic semidefinite programming problems, Computational Optimization and Applications, 57.1, pp. 45–69, 2014.

[6] J. Povh, F. Rendl and A. Wiegele, A boundary point method to solve semidefinite programs, Computing, 78.3, pp. 277–286, 2006.

[7] M. Stingl, M. Kočvara and G. Leugering, A sequential convex semidefinite programming algorithm with an application to multiple-load free material optimization, SIAM Journal on Optimization, 20.1, pp. 130–155, 2009.

[8] J. F. Sturm, Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, Optimization Methods and Software, 11.1–4, pp. 625–653, 1999.

[9] K. C. Toh, M. J. Todd, and R. H. Tütüncü, SDPT3-a MATLAB software package for semidefinite programming, version 1.3, Optimization Methods and Software, 11.1–4, pp. 545–581, 1999.

[10] M. Yamashita, K. Fujisawa and M. Kojima, Implementation and evaluation of SDPA 6.0 (semidefinite programming algorithm 6.0), Optimization Methods and Software, 18.4, pp. 491–505, 2003.