令和 4 年度　学士論文

taitoru
タイトル

東京工業大学　情報理工学院　数理・計算科学系
学籍番号　19B31048
脇田康平
指導教員 山下 真 教授

2022 年 2 月 27 日

目次

1　はじめに

本論文の構成は以下である。第 2 節では、二次錐計画問題 (SOCP) に関する前提知識として二次錐計画問題と内点法アルゴリズム、中心パスなどについて説明をする。第 3 節では、緩和法とペナルティ法に基づく変形について説明する。第 5 節では、数値実験を行った結果を示し、各手法の評価を行う。第 6 節で、まとめを行う。

1.1　記法

この論文で扱う記法を述べる。

- $\|\cdot\|$ はユークリッドノルムとする。
- 行列 $\boldsymbol{A} \in \mathbb{R}^{k\times m}, \boldsymbol{B} \in \mathbb{R}^{k\times n}$ を横方向につなげてできる新たな行列 $\boldsymbol{C} \in \mathbb{R}^{k\times(m+n)}$ を $\boldsymbol{C} = (\boldsymbol{A}, \boldsymbol{B})$ と表記する。
- 行列 $\boldsymbol{A} \in \mathbb{R}^{m\times k}, \boldsymbol{B} \in \mathbb{R}^{n\times k}$ を縦方向につなげてできる新たな行列 $\boldsymbol{C} \in \mathbb{R}^{(m+n)\times k}$ を $\boldsymbol{C} = (\boldsymbol{A}; \boldsymbol{B})$ と表記する。
- 上付き添え字 T は行列やベクトルの転置を示す。
- $\boldsymbol{x}^T\boldsymbol{y}$ はベクトル $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ の内積 $\sum_{i=1}^n x_i y_i$ である。
- 2 つのベクトル $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ に対して、$\boldsymbol{x} \circ \boldsymbol{y}$ を次のように定義する。

$$\boldsymbol{x} \circ \boldsymbol{y} = \begin{pmatrix} \boldsymbol{x}^T\boldsymbol{y} \\ x_0 y_1 + y_0 x_1 \\ \vdots \\ x_0 y_n + y_0 x_n \end{pmatrix}$$

- 2 つの行列 $\boldsymbol{A}, \boldsymbol{B}$ に対して $\boldsymbol{A} \oplus \boldsymbol{B} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B} \end{pmatrix}$
- n 次元のベクトル $\boldsymbol{e} \in \mathbb{R}^n$ かつ、1 番目の要素のみが 1 で、その他の要素は全て 0 であるベクトルを $\boldsymbol{e}_n = (1; \boldsymbol{0})$ と表す。
- 対角成分以外は 0 で、対角成分は 1 行目が 1, それ以外が -1 である行列を $\boldsymbol{R}_n \in \mathbb{R}^{n\times n}$ とする。$\boldsymbol{R}_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{pmatrix}$

2　前提知識

この章では、MISOCP とヒューリスティクスに関する前提知識について述べる。

2.1　混合整数二次錐計画問題

2.1.1　二次錐計画問題

本論文で扱う SOCP の標準形は、以下のように定義され、これを主問題と呼ぶ。

$$\begin{array}{rcl} \min & : & \boldsymbol{c}_1^T\boldsymbol{x}_1 + \cdots + \boldsymbol{c}_m^T\boldsymbol{x}_m \\ \text{subject to} & : & \boldsymbol{A}_1\boldsymbol{x}_1 + \cdots + \boldsymbol{A}_m\boldsymbol{x}_m = \boldsymbol{b} \\ & & \boldsymbol{x}_i \in \mathcal{K}_i \ (\text{for } i = 1, \ldots, m) \end{array} \qquad (\mathcal{P})$$

ここで、$i = 1, \ldots, m$ に対して $\boldsymbol{A}_i \in \mathbb{R}^{n \times n_i}, \boldsymbol{b} \in \mathbb{R}^n, \boldsymbol{c}_i \in \mathbb{R}^{n_i}$ が定数で $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$ が変数である。また、$\mathcal{K}_i$ は n_i 次元の二次錐である。また、$(\mathcal{P})$ において、$\boldsymbol{e} = (\boldsymbol{e}_{n_1}, \ldots, \boldsymbol{e}_{n_m})$, $\boldsymbol{R} = \boldsymbol{R}_{n_1} \oplus \cdots \oplus \boldsymbol{R}_{n_m}$ とする。

一般的に n 次元の二次錐 $\mathcal{K} \subset \mathbb{R}^n$ は、以下で定義される。

$$\mathcal{K} = \left\{ \boldsymbol{x} = \begin{pmatrix} x_0 \\ \boldsymbol{x}_1 \end{pmatrix} \;\middle|\; x_0 \in \mathbb{R}, \boldsymbol{x}_1 \in \mathbb{R}^{n-1}, x_0 \geq \|\boldsymbol{x}_1\| \right\}$$

二次錐 $\mathcal{K}$ に対して、境界と内部は次のように定義される。

$$\text{int } \mathcal{K} = \{\boldsymbol{x} \in \mathcal{K} \mid x_0 > \|\boldsymbol{x}_1\|\}, \quad \text{bd } \mathcal{K} = \{\boldsymbol{x} \in \mathcal{K} \mid x_0 = \|\boldsymbol{x}_1\|\}$$

$(\mathcal{P})$ に対し、$\boldsymbol{x}$ が等式制約 $\boldsymbol{A}_1\boldsymbol{x}_1 + \cdots + \boldsymbol{A}_m\boldsymbol{x}_m = \boldsymbol{b}$ および $\boldsymbol{x}_i \in \mathcal{K}_i$ (for $i = 1, \ldots, m$) を満たすとき実行可能解と呼ぶ。さらに実行可能解であり内部に存在する ($\boldsymbol{x}_i \in \text{int } \mathcal{K}_i$ for $i = 1, \ldots, m$) とき、実行可能内点と呼ぶ。

$(\mathcal{P})$ を主問題とするとき、対応する双対問題は以下の形式で与えられる。

$$\begin{array}{rcl} \max & : & \boldsymbol{b}^T\boldsymbol{y} \\ \text{subject to} & : & \boldsymbol{A}_i^T\boldsymbol{y} + \boldsymbol{z}_i = \boldsymbol{c}_i \ (\text{for } \ i = 1, \ldots, m) \\ & & \boldsymbol{z}_i \in \mathcal{K}_i \ (\text{for } \ i = 1, \ldots, m) \end{array} \qquad (\mathcal{D})$$

主問題と同様に、$(\mathcal{D})$ に対して $(\boldsymbol{y}, \boldsymbol{z})$ が等式制約 $\boldsymbol{A}_i^T\boldsymbol{y} + \boldsymbol{z}_i = \boldsymbol{c}_i, \boldsymbol{z}_i \in \mathcal{K}_i$ (for $i = 1, \ldots, m$) を満たすとき実行可能解と呼び、さらに $\boldsymbol{z}_i \in \text{int } \mathcal{K}_i$ (for $i = 1, \ldots, m$) を満たすとき実行可能内点と呼ぶ。

2.1.2　混合整数二次錐計画問題

線形計画問題 (LP) は多くの実用上の応用問題を定式化可能であるが、SOCP は LP を含むさらに大きな問題のクラスである。この包含関係を見るために、以下の標準形で与えられる LP が SOCP で定式化できることを示す。

$$\begin{array}{rcl} \min & : & \sum_{i=1}^{m} c_i x_i \\ \text{subject to} & : & \sum_{i=1}^{m} \boldsymbol{a}_i x_i = \boldsymbol{b} \\ & & x_i \geq 0 \ (i = 1, \ldots, m) \end{array}$$

ここで、$c_i \in \mathbb{R}\ (i = 1, \ldots, m), \boldsymbol{a}_i \in \mathbb{R}^{n_i}\ (i = 1, \ldots, m)$ が定数で $x_i \in \mathbb{R}\ (i = 1, \ldots, m)$ が変数である。特に、各変数は非負制約が課されている。一方で SOCP における二次錐制約は

$$x_0 \geq \|\boldsymbol{x}_1\|\ ,\ \boldsymbol{x}_1 \in \mathbb{R}^{n-1}$$

であるが、ここで $n = 1$ の場合、つまり 1 次元の二次錐を考えると右辺の $\|\boldsymbol{x}_1\|$ が 0 であり、$x_0 \geq 0$ となり実質的に非負制約である。したがって、LP における非負制約を 1 次元の二次錐として捉えることで、一般に LP は SOCP は帰着可能である。

第 1 節でも触れたとおり SOCP は理学・工学の幅広い分野へと応用されているが、その数例をここで紹介する。

具体例 1)：<u>ノルム最小化問題</u>

ノルム最小化問題は次のように定式化される。

$$\begin{array}{rcll} \min & : & \sum_{i=1}^{n} \|\overline{\boldsymbol{v}}_i\| & \\ \text{subject to} & : & \boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i = \overline{\boldsymbol{v}}_i & (i = 1, \ldots, n) \end{array}$$

このままだと、二次錐制約が入っておらず、SOCP の形になっていない。そこで、$\boldsymbol{v}_i = (v_{i0}; \overline{\boldsymbol{v}}_i)$ とし、n 次元の二次錐 $\mathcal{K}$ に対して、$\boldsymbol{v}_i \in \mathcal{K}\ (i = 1, \ldots, n)$ をいう制約を追加することで、$v_{i0} \geq \sum_{i=1}^{n} \|\overline{\boldsymbol{v}}_i\|$ より、目的関数 $\sum_{i=1}^{n} v_{i0}$ を最小化する次の SOCP に変形することができる。

$$\begin{array}{rcll} \min & : & \sum_{i=1}^{n} v_{i0} & \\ \text{subject to} & : & \boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i = \overline{\boldsymbol{v}}_i & (i = 1, \ldots, n) \\ & & \boldsymbol{v}_i \in \mathcal{K} & (i = 1, \ldots, n) \end{array}$$

この問題は、インパルス性雑音除去のためのネットワーク最適化問題や、シュタイナー木問題、最適位置問題などを解く際に用いられる [**?**]。

具体例 2)：<u>最大ノルム最小化問題</u>

最大ノルム最小化問題は次のように定式化される。

$$\begin{array}{rcll} \min & : & \max\limits_{i=1,\ldots,m} \|v_i\| & \\ \text{subject to} & : & \boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i = \boldsymbol{v}_i & (i = 1, \ldots, n) \end{array}$$

ここで、t を全ての i に対して $t \geq \|v_i\|$ を満たす変数とおくと、この問題は t の値を最小化する問題と等しい。よって、次のような SOCP に変形できる。

$$\begin{array}{rcll} \min & : & t & \\ \text{subject to} & : & \boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i = \boldsymbol{v}_i & (i = 1, \ldots, n) \\ & & (t; \boldsymbol{v}_i) \in \mathcal{K} & (i = 1, \ldots, n) \end{array}$$

この問題は、平面状における全方向性アンテナの感度の最小化問題や、周波数近似フィルタの誤差の最小化問題などを解く際に用いられる [**?**]。

2.2 分枝限定法

二次錐 $\mathcal{K}$ の重要な性質として、自己双対錐であることが挙げられる。つまり、双対錐を

$$\mathcal{K}^* := \{\boldsymbol{z} \mid \boldsymbol{x}^T\boldsymbol{z} \geq 0 \text{ for } \boldsymbol{x} \in \mathcal{K}\}$$

により定義したときに、以下の定理が成り立つ。

定理 1. $\mathcal{K}^* = \mathcal{K}$ が成り立つ。

Proof. $\boldsymbol{z} \in \mathcal{K}$ ならば、$z_0 \geq \|\boldsymbol{z}_1\|$ より、$\boldsymbol{x} \in \mathcal{K}$ に対して

$$\boldsymbol{x}^T\boldsymbol{z} = z_0x_0 + \boldsymbol{z}_1^T\boldsymbol{x}_1 \geq z_0x_0 - \|\boldsymbol{z}_1\|\|\boldsymbol{x}_1\| \geq 0$$

となり、$\boldsymbol{z} \in \mathcal{K}^*$ が示される。つまり $\mathcal{K} \subset \mathcal{K}^*$ が成り立つ。

逆に $\boldsymbol{z} \in \mathcal{K}^*$ ならば、$\boldsymbol{x} \in \mathcal{K}$ に対して $\boldsymbol{x}^T\boldsymbol{z} \geq 0$ であることから $x_0 = 1, \boldsymbol{x}_1 = \boldsymbol{0}$ とすれば

$$\boldsymbol{x}^T\boldsymbol{z} = z_0x_0 + \boldsymbol{z}_1^T\boldsymbol{x}_1 = z_0 \geq 0$$

がわかる。さらに、$x_0 = z_0, \boldsymbol{x}_1 = -\boldsymbol{z}_1$ とすれば

$$\boldsymbol{x}^T\boldsymbol{z} = z_0x_0 + \boldsymbol{z}_1^T\boldsymbol{x}_1 = z_0^2 - \|\boldsymbol{z}_1\|^2 \geq 0$$

となり $\boldsymbol{z} \in \mathcal{K}$ となる。つまり $\mathcal{K}^* \subset \mathcal{K}$ が成り立つ。したがって $\mathcal{K} = \mathcal{K}^*$ を得る。 □

この定理から、以下の系 2 が導出される。この系 2 は弱双対定理の証明に利用される。

系 2. $\boldsymbol{x} \in \mathcal{K}, \boldsymbol{z} \in \mathcal{K}$ であれば、$\boldsymbol{x}^T\boldsymbol{z} \geq 0$ である。

Proof. 定理 1 より、$\boldsymbol{z} \in \mathcal{K}$ から $\boldsymbol{z} \in \mathcal{K}^*$ である。したがって、$\boldsymbol{x} \in \mathcal{K}$ と $\mathcal{K}^*$ の定義から $\boldsymbol{x}^T\boldsymbol{z} \geq 0$ である。 □

2.3 焼きなまし法

多くの既存研究において、$(\mathcal{P})$, $(\mathcal{D})$ に対して以下の仮定を置いている。

仮定 3. (i) 行列 $\boldsymbol{A} = (\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_m)$ の行ベクトルは線形独立である。
(ii) 主問題 $(\mathcal{P})$ と双対問題 $(\mathcal{D})$ のそれぞれの実行可能領域は空集合ではない。

仮定 (i) は、探索方向を求める際の線形方程式 (**??**) が解を持つために必要な条件である。ただし、仮定 (i) が満たされない場合でも、主問題 $(\mathcal{P})$ の制約を線形方程式系とみなしてガウスの消去法により冗長な制約を削除することにより、一般性を失うことなく仮定 (i) を仮定することが可能である。仮定 (ii) は、実行可能領域が空集合ならば、最適解のなす集合も空集合である。数値実験の際に安定して内点法で解を求める事ができるためにも必要である。本論文でも、仮定 (i) および仮定 (ii) を仮定する。

SOCP における理論的な性質として、$(\mathcal{P})$ と $(\mathcal{D})$ の関係性は重要である。まず最初の性質として、主問題の目的関数値は双対問題の目的関数値の上界となっており、同様に双対問題の目的関数値は主問題の目的関数値の下界である。このことは、以下の弱双対定理によって示される。

定理 4 (弱双対定理). $\boldsymbol{x} = (\boldsymbol{x}_1; \ldots; \boldsymbol{x}_m)$ を $(\mathcal{P})$ の任意の実行可能解、$(\boldsymbol{y}, \boldsymbol{z})$ ただし $\boldsymbol{z} = (\boldsymbol{z}_1; \ldots; \boldsymbol{z}_m)$ を $(\mathcal{D})$ の任意の実行可能解とする。このとき、以下が成り立つ。

$$\sum_{i=1}^{m} \boldsymbol{c}_i^T \boldsymbol{x}_i - \boldsymbol{b}^T \boldsymbol{y} = \sum_{i=1}^{m} \boldsymbol{z}_i^T \boldsymbol{x}_i \geq 0$$

Proof. $(\mathcal{P})$ の制約から $\sum_{i=1}^{m} \boldsymbol{A}_i \boldsymbol{x}_i = \boldsymbol{b}$ であり、$(\mathcal{D})$ の制約から $\boldsymbol{A}_i^T \boldsymbol{y} + \boldsymbol{z}_i = \boldsymbol{c}_i$ (for $i = 1, \ldots, m$) であることから

$$\sum_{i=1}^{m} \boldsymbol{c}_i^T \boldsymbol{x}_i - \boldsymbol{b}^T \boldsymbol{y} = \sum_{i=1}^{m} (\boldsymbol{A}_i^T \boldsymbol{y} + \boldsymbol{z}_i)^T \boldsymbol{x}_i - \boldsymbol{b}^T \boldsymbol{y} = \boldsymbol{y}^T \boldsymbol{b} + \sum_{i=1}^{m} \boldsymbol{z}_i^T \boldsymbol{x}_i - \boldsymbol{b}^T \boldsymbol{y} = \sum_{i=1}^{m} \boldsymbol{z}_i^T \boldsymbol{x}_i$$

である。ここで、系 2 より $\boldsymbol{x}_i \in \mathcal{K}_i, \boldsymbol{z}_i \in \mathcal{K}_i$ から $\boldsymbol{z}_i^T \boldsymbol{x}_i \geq 0$ より $\sum_{i=1}^{m} \boldsymbol{z}_i^T \boldsymbol{x}_i \geq 0$ である。 □

ここで、$\sum_{i=1}^{m} \boldsymbol{x}_i^T \boldsymbol{z}_i$ は双対ギャップと呼ばれ、正の値となることもあるが、内点の仮定を置くと双対ギャップが 0 になることが知られている。これは、次の定理から示される。

定理 5 (強双対定理 [?]). $(\mathcal{P})$,$(\mathcal{D})$ がそれぞれ実行可能内点を持つとき、それぞれの問題に最適解 $(\boldsymbol{x}^*) = (\boldsymbol{x}_1^*; \ldots; \boldsymbol{x}_m^*)$ と $(\boldsymbol{y}^*, \boldsymbol{z}^*)$ ただし $\boldsymbol{z}^* = (\boldsymbol{z}_1^*; \ldots; \boldsymbol{z}_m^*)$ が存在して、それらの最適値は一致する。つまり、次の式が成り立つ。

$$\sum_{i=1}^{m} \boldsymbol{c}_i^T \boldsymbol{x}_i^* = \boldsymbol{b}^T \boldsymbol{y}^* \quad (i.e. \quad \sum_{i=1}^{m} \boldsymbol{z}_i^{*T} \boldsymbol{x}_i^* = 0)$$

3 問題設定と定式化

前節でみたように内点の存在は内点法の計算で重要であるが、電力潮流問題 [?] などでは実際に内点を持たない SOCP が現れる。そこで、内点がないという制約付きの二次錐計画問題を数値的に安定して求解可能とする方法について考える。

3.1 問題設定

内点を持たない SOCP に対する主双対内点法の数値的安定性を数値実験で比較するために、この小節では、そのような内点を持たない SOCP の生成方法を検討する。

まず、以下の SOCP を考える。

$$\begin{array}{rcl} \min & : & \boldsymbol{f}^T \boldsymbol{x} \\ \text{subject to} & : & \boldsymbol{c}_i^T \boldsymbol{x} + d_i \geq \|\boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i\| \ (i = 1, \ldots, m) \\ & & \boldsymbol{F}\boldsymbol{x} = \boldsymbol{g} \end{array}$$

ここで、$\boldsymbol{A}_i \in \mathbb{R}^{n\times n_i}, \boldsymbol{b}_i \in \mathbb{R}^n, \boldsymbol{c}_i \in \mathbb{R}^{n_i}, d_i \in \mathbb{R}, \boldsymbol{f} \in \mathbb{R}^{n_i}, \boldsymbol{F} \in \mathbb{R}^{p\times n_i}, \boldsymbol{g} \in \mathbb{R}^p$ が定数であり、$\boldsymbol{x} \in \mathbb{R}^{n_i}$ が決定変数である。この問題は、不等式制約が

$$\begin{pmatrix} \boldsymbol{c}_i^T\boldsymbol{x} + d_i \\ \boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i \end{pmatrix} = \begin{pmatrix} \boldsymbol{c}_i^T \\ \boldsymbol{A}_i \end{pmatrix}\boldsymbol{x} + \begin{pmatrix} d_i \\ \boldsymbol{b}_i \end{pmatrix} \in \mathcal{K}$$

と等しいので、$(\mathcal{P})$ で出ている二次錐制約に変形することができることから SOCP である。特に、$\boldsymbol{A}_i \in \mathbb{R}^{n\times n_i}, \boldsymbol{b}_i \in \mathbb{R}^n, \boldsymbol{c}_i \in \mathbb{R}^{n_i}, \boldsymbol{F} \in \mathbb{R}^{p\times n_i}$ と、ある $\overline{\boldsymbol{x}} \in \mathbb{R}^{n_i}$ に対して

$$\begin{aligned} d_i &= \|\boldsymbol{A}_i\overline{\boldsymbol{x}} + \boldsymbol{b}_i\| - \boldsymbol{c}_i^T\overline{\boldsymbol{x}} + 1 \\ \boldsymbol{g} &= \boldsymbol{F}\overline{\boldsymbol{x}} \end{aligned}$$

のようにデータを入力することで、実行可能な内点が存在する問題を生成することが可能である。

この問題を変形することで、実行可能内点が存在しないような SOCP を生成する。具体的には、$\overline{m}$ 本の二次錐制約と $\overline{m}$ 本の等式制約を付け加えた以下のような SOCP を考える。

$$\begin{array}{rcl} \min & : & \boldsymbol{f}^T\boldsymbol{x} \\ \text{subject to} & : & \boldsymbol{c}_i^T\boldsymbol{x} + d_i \geq \|\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i\| \ (i = 1, \ldots, m) \\ & & \overline{\boldsymbol{c}}_i^T\boldsymbol{x} + \overline{d}_i \geq \|\overline{\boldsymbol{A}}_i\boldsymbol{x} + \overline{\boldsymbol{b}}_i\| \ (i = 1, \ldots, \overline{m}) \\ & & (\overline{\boldsymbol{c}}_i^T\overline{\boldsymbol{x}} + \overline{d}_i)(\overline{\boldsymbol{c}}_i^T\boldsymbol{x} + \overline{d}_i) - (\overline{\boldsymbol{A}}_i\overline{\boldsymbol{x}} + \overline{\boldsymbol{b}}_i)^T(\overline{\boldsymbol{A}}_i\boldsymbol{x} + \overline{\boldsymbol{b}}_i) = 0 \ (i = 1, \ldots, \overline{m}) \\ & & \boldsymbol{F}\boldsymbol{x} = \boldsymbol{g} \end{array} \qquad (\mathcal{P}_o)$$

この問題 $(\mathcal{P}_o)$ において $\overline{d}_i$ は、$\overline{\boldsymbol{A}}_i \in \mathbb{R}^{n\times n_i}, \overline{\boldsymbol{b}}_i \in \mathbb{R}^n, \overline{\boldsymbol{c}}_i \in \mathbb{R}^{n_i}, \overline{\boldsymbol{x}} \in \mathbb{R}^{n_i}$ に対して

$$\overline{d}_i = -\overline{\boldsymbol{c}}_i^T\overline{x} + \|\overline{\boldsymbol{A}}_i\overline{\boldsymbol{x}} + \overline{\boldsymbol{b}}_i\|$$

を満たすように設定する。

このとき、

$$\begin{pmatrix} \overline{\boldsymbol{c}}_i^T \\ \overline{\boldsymbol{A}}_i \end{pmatrix}\overline{\boldsymbol{x}} + \begin{pmatrix} \overline{d}_i \\ \overline{\boldsymbol{b}}_i \end{pmatrix} \notin \text{int } \mathcal{K} \ , \ \begin{pmatrix} \overline{\boldsymbol{c}}_i^T \\ \overline{\boldsymbol{A}}_i \end{pmatrix}\overline{\boldsymbol{x}} + \begin{pmatrix} \overline{d}_i \\ \overline{\boldsymbol{b}}_i \end{pmatrix} \in \text{bd } \mathcal{K}$$

である。さらに

$$\mathcal{H} = \{\boldsymbol{x} \mid (\overline{\boldsymbol{c}}_i^T\overline{\boldsymbol{x}} + \overline{d}_i)(\overline{\boldsymbol{c}}_i^T\boldsymbol{x} + \overline{d}_i) - (\overline{\boldsymbol{A}}_i\overline{\boldsymbol{x}} + \overline{\boldsymbol{b}}_i)^T(\overline{\boldsymbol{A}}_i\boldsymbol{x} + \overline{\boldsymbol{b}}_i) = 0\}$$

とすると、$\mathcal{K} \cap \mathcal{H} \subset \text{bd } \mathcal{K}$ が成り立つ。よって、この問題は実行可能内点を持たないことがわかる。

この SOCP は、第**??**節で議論したように内点法で解こうとすると数値的に不安定になりやすく、これを解消する必要がある。解消する方法として、本論文ではペナルティ法 [**?**], [**?**] と緩和法 [**?**] を検討する。

3.2 MISOCP として定式化

この方法では、$(\mathcal{P}_o)$ の二次錐制約に対して、ペナルティ係数 ξ_i を加えることで不等式制約を緩めることを考える。一方で、ペナルティ項を目的関数に追加して、ペナルティも最小化することで $(\mathcal{P}_o)$ になるべく近い解を求めるように問題を変形する。

具体的には、$\boldsymbol{c}_i^T\boldsymbol{x}+d_i \geq \|\boldsymbol{A}_i\boldsymbol{x}+\boldsymbol{b}_i\|$ と $\overline{\boldsymbol{c}}_i^T\boldsymbol{x}+\overline{d}_i \geq \|\overline{\boldsymbol{A}}_i\boldsymbol{x}+\overline{\boldsymbol{b}}_i\|$ の制約を、それぞれ $\xi_i \geq 0$ と $\overline{\xi}_i \geq 0$ を入れることで緩和する。さらに、$\xi_i, \overline{\xi}_i$ ができるだけ 0 に近づけるように目的関数にペナルティ項を追加する。つまり、以下の問題の求解を行う。

$$
\begin{array}{rcl}
\min & : & \boldsymbol{f}^T\boldsymbol{x}+\sum_{i=1}^{m} P_i\xi_i+\sum_{i=1}^{\overline{m}} \overline{P}_i\overline{\xi}_i \\
\text{subject to} & : & \boldsymbol{c}_i^T\boldsymbol{x}+d_i+\xi_i \geq \|\boldsymbol{A}_i\boldsymbol{x}+\boldsymbol{b}_i\| \ (i=1,\ldots,m) \\
 & & \overline{\boldsymbol{c}}_i^T\boldsymbol{x}+\overline{d}_i+\overline{\xi}_i \geq \|\overline{\boldsymbol{A}}_i\boldsymbol{x}+\overline{\boldsymbol{b}}_i\| \ (i=1,\ldots,\overline{m}) \\
 & & (\overline{\boldsymbol{c}}_i^T\overline{\boldsymbol{x}}+\overline{d}_i)(\overline{\boldsymbol{c}}_i^T\boldsymbol{x}+\overline{d}_i)-(\overline{\boldsymbol{A}}_i\overline{\boldsymbol{x}}+\overline{\boldsymbol{b}}_i)^T(\overline{\boldsymbol{A}}_i\boldsymbol{x}+\overline{\boldsymbol{b}}_i)=0 \ (i=1,\ldots,\overline{m}) \\
 & & \boldsymbol{F}\boldsymbol{x}=\boldsymbol{g} \\
 & & \xi_i \geq 0 \ (i=1,\ldots,m), \ \overline{\xi}_i \geq 0 \ (i=1,\ldots,\overline{m})
\end{array}
\qquad (\mathcal{P}_p)
$$

ここで、$P_1,\ldots,P_m,\overline{P}_1,\ldots,\overline{P}_{\overline{m}}$ はペナルティの重みで、十分大きな正の値とする。入力データにもよるが、各データを $[0,1]$ 区間で乱数で生成した場合には、大体の目安として 10^3 や 10^4 などの値が入る。

このペナルティ法は、新たなペナルティ項を追加したことで実行可能領域が広がるため、必ず実行可能内点を持つという性質を持つ。具体的には、$\xi_i, \overline{\xi}_i$ を

$$
\xi_i \geq \|\boldsymbol{A}_i\overline{\boldsymbol{x}}+\boldsymbol{b}_i\|-(\boldsymbol{c}_i^T\overline{\boldsymbol{x}}+\boldsymbol{d}_i)+1 \ , \ \overline{\xi}_i \geq \|\overline{\boldsymbol{A}}_i\overline{\boldsymbol{x}}+\overline{\boldsymbol{b}}_i\|-(\overline{\boldsymbol{c}}_i^T\overline{\boldsymbol{x}}+\overline{\boldsymbol{d}}_i)+1
$$

を満たすようにとる。すると、$\boldsymbol{x}=\overline{\boldsymbol{x}}$ は、不等式制約

$$
\boldsymbol{c}_i^T\boldsymbol{x}+d_i+\xi_i > \|\boldsymbol{A}_i\boldsymbol{x}+\boldsymbol{b}_i\| \ (i=1,\ldots,m), \quad \overline{\boldsymbol{c}}_i^T\boldsymbol{x}+\overline{d}_i+\overline{\xi}_i > \|\overline{\boldsymbol{A}}_i\boldsymbol{x}+\overline{\boldsymbol{b}}_i\| \ (i=1,\ldots,\overline{m})
$$

を満たしているので $\overline{\boldsymbol{x}}$ は内点であり、かつ等式制約 $\boldsymbol{F}\boldsymbol{x}=\boldsymbol{g}$ および $(\overline{\boldsymbol{c}}_i^T\overline{\boldsymbol{x}}+\overline{d}_i)(\overline{\boldsymbol{c}}_i^T\boldsymbol{x}+\overline{d}_i)-(\overline{\boldsymbol{A}}_i\overline{\boldsymbol{x}}+\overline{\boldsymbol{b}}_i)(\overline{\boldsymbol{A}}_i\boldsymbol{x}+\overline{\boldsymbol{b}}_i)=0$ を満たしているので実行可能でもある。よって、実際に $(\mathcal{P}_p)$ の実行可能内点として $\overline{\boldsymbol{x}}$ をとることができる。この効果によって、内点法で安定して解くことができると期待できる。

3.3 non

緩和法は、内点を持たないような不等式制約に着目して、不等式制約を等式制約と錐制約に分解することで、数値的に安定して求解しようとする方法である。例えば、各反復で錐制約については内点である必要があるが、infeasible interior-point method では等式制約については必ずしも厳密に満たさなくても探索方向の計算が可能である。具体的には、$(\mathcal{P}_o)$ の二次錐の制約に関して、新しく変数を入れることで等式制約と二次錐 $\mathcal{K}=\left\{\boldsymbol{z}=\begin{pmatrix} z_0 \\ \boldsymbol{z}_1 \end{pmatrix} : z_0 \geq \|\boldsymbol{z}_1\|\right\}$ に分解する。実際には、

$$
\begin{pmatrix} \boldsymbol{c}_i^T\boldsymbol{x}+d_i \\ \boldsymbol{A}_i\boldsymbol{x}+\boldsymbol{b}_i \end{pmatrix} \in \mathcal{K}
$$

で表現されている二次錐制約に対して、新しい変数 $\boldsymbol{y}_i$ を導入し等式制約と二次錐制約に分ける。

$$
\begin{pmatrix} \boldsymbol{c}_i^T\boldsymbol{x}+d_i \\ \boldsymbol{A}_i\boldsymbol{x}+\boldsymbol{b}_i \end{pmatrix} \in \mathcal{K} \iff \boldsymbol{y}_i=\begin{pmatrix} \boldsymbol{c}_i^T\boldsymbol{x}+d_i \\ \boldsymbol{A}_i\boldsymbol{x}+\boldsymbol{b}_i \end{pmatrix}, \ \boldsymbol{y}_i \in \mathcal{K}
$$

これによって、数値誤差が等式制約 $\boldsymbol{y}_i = \begin{pmatrix} \boldsymbol{c}_i^T\boldsymbol{x} + d_i \\ \boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i \end{pmatrix}$ の部分に出るようになるので、安定して解くことができるようになると期待できる。したがって、解く問題は以下のようになる。

$$
\begin{aligned}
\min \quad &: \quad \boldsymbol{f}^T\boldsymbol{x} \\
\text{subject to} \quad &: \quad \boldsymbol{y}_i = \begin{pmatrix} \boldsymbol{c}_i^T\boldsymbol{x} + d_i \\ \boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i \end{pmatrix} \ (i = 1, \ldots, m) \\
&\quad\ \ \overline{\boldsymbol{y}}_i = \begin{pmatrix} \overline{\boldsymbol{c}}_i^T\boldsymbol{x} + \overline{d}_i \\ \overline{\boldsymbol{A}}_i\boldsymbol{x} + \overline{\boldsymbol{b}}_i \end{pmatrix} \ (i = 1, \ldots, \overline{m}) \qquad (\mathcal{P}_r) \\
&\quad\ \ (\overline{\boldsymbol{c}}_i^T\overline{\boldsymbol{x}} + \overline{d}_i)(\overline{\boldsymbol{c}}_i^T\boldsymbol{x} + \overline{d}_i) - (\overline{\boldsymbol{A}}_i\overline{\boldsymbol{x}} + \overline{\boldsymbol{b}}_i)^T(\overline{\boldsymbol{A}}_i\boldsymbol{x} + \overline{\boldsymbol{b}}_i) = 0 \ (i = 1, \ldots, \overline{m}) \\
&\quad\ \ \boldsymbol{F}\boldsymbol{x} = \boldsymbol{g} \\
&\quad\ \ \boldsymbol{y}_i \ (i = 1, \ldots, m), \ \overline{\boldsymbol{y}}_i \ (i = 1, \ldots, \overline{m}) \in \mathcal{K}
\end{aligned}
$$

この緩和法では、数値誤差の影響が錐 ($\mathcal{K}$) から等式条件のほうに強くなるため、数値的には安定して解きやすくなる。しかし、追加する変数や制約の数が多くなってしまう。例えば、元の問題 ($\mathcal{P}_o$) の制約の本数が $m+2\overline{m}+1$ 本であるのに対して、この問題 ($\mathcal{P}_p$) の制約の本数は $2m+3\overline{m}+1$ 本である。変数は、元の問題 ($\mathcal{P}_o$) の制約の本数が m 本であるのに対して、この問題 ($\mathcal{P}_p$) の制約の本数は $n + m\sum_{i=1}^{m}(n_i+1) + \overline{m}\sum_{i=1}^{\overline{m}}(n_i+1)$ 本である。これからわかるように、n_i の数値が大きいほど、サイズが大きい場合に変数の数が急激に多くなり、計算に必要な変数と制約の数が他の 2 手法に比べてかなり多くなる。このため、最適解の求解に要する時間が増えてしまうというデメリットもある。

4　提案手法

5　数値実験

この節では、前節で導入した $(\mathcal{P}_o)$,$(\mathcal{P}_p)$,$(\mathcal{P}_r)$ の 3 つのモデルで内点のない SOCP を求解し、数値的に比較を行う。ここでは、SOCP を解くソルバーとして SDPT3-4.0 [**?**] を用いた。詳しい数値実験の実行環境は表 1 の通りである。

表 1　実行環境

プロセッサ	1.6 GHz デュアルコア Intel Core i5
メモリ	8 GB 2133 MHz LPDDR3
OS	macOS Big Sur バージョン 11.6
実装言語	MATLAB_R2021b
ソルバー	SDPT3-4.0

5.1　定数の設定

今回使用した SDPT3 では、次のような形式が入力形式であるため、この形式に沿うように入力をする必要がある [**?**]。

$$
\begin{aligned}
\min \quad : \quad & \sum_{j=1}^{n_s}\left[\langle c_j^s, x_j^s\rangle - v_j^s \log\det(x_j^s)\right] \quad + \sum_{i=1}^{n_q}\left[\langle c_i^q, x_i^q\rangle - v_i^q \log\det(x_i^q)\right] \\
& + \langle c^l, x^l\rangle \; - \; \sum_{k=1}^{n_l} v_k^l \log x_k^l \qquad\qquad + \langle c^u, x^u\rangle \\
\text{subject to} \quad : \quad & \sum_{j=1}^{n_s} A_j^s(x_j^s) \quad + \sum_{j=1}^{n_q} A_j^q(x_j^q) \qquad + A^l x^l + \; A^u x^u = b \\
& x_j^s \in \mathcal{K}_s^{s_j} \;\; \forall j \;,\; x_i^q \in \mathcal{K}_q^{q_i} \;\; \forall i, \qquad x^l \in \mathcal{K}_l^{n_l} \;,\; x^u \in \mathbb{R}^{n_u}
\end{aligned}
$$

この問題に対応する双対問題は以下の問題である。

$$
\begin{aligned}
\min \quad : \quad & \sum_{j=1}^{n_s}\left[\langle c_j^s, x_j^s\rangle - v_j^s \log\det(x_j^s)\right] \quad + \sum_{i=1}^{n_q}\left[\langle c_i^q, x_i^q\rangle - v_i^q \log\det(x_i^q)\right] \\
& + \langle c^l, x^l\rangle \; - \; \sum_{k=1}^{n_l} v_k^l \log x_k^l \qquad\qquad + \langle c^u, x^u\rangle \\
\text{subject to} \quad : \quad & (A_j^s)^T y \;+\; z_j^s \;=\; c_j^s, \quad z_j^s \in \mathcal{K}_s^{s_j} \quad (j = 1, ..., n_s) \\
& (A_i^q)^T y \;+\; z_i^q \;=\; c_i^q, \quad z_i^q \in \mathcal{K}_q^{q_i} \quad (i = 1, ..., n_q) \\
& (A^l)^T y \;+\; z^l \;=\; c^l, \quad z^l \in \mathcal{K}_l^{n_l} \\
& (A^u)^T y \;\;\;\;\;\;\;\; =\; c^u, \quad y \in \mathbb{R}^m
\end{aligned}
$$

ここで、$x = (x_1^s, \ldots, x_{n_s}^s, x_1^q, \ldots, x_{n_q}^q, x^l)$, ただし $x_1^s, \ldots, x_{n_s}^s$ は対称行列, $x_1^q, \ldots, x_{n_q}^q, x^l$ はベクトルである。

一般に K を自己双対なユークリッド空間上の閉凸錐、$A : X \to \mathbb{R}^m$ を線形写像、A^* を A の随伴行列、$b \in \mathbb{R}^m$、$c \in X$ とする。これらを用いるとソルバーの入力標準形は、以下の形式で簡潔に表すこともできる。

$$\begin{array}{rcl@{\qquad}rcl} \min & : & \langle \boldsymbol{c}, \boldsymbol{x} \rangle & \max & : & \boldsymbol{b}^T \boldsymbol{y} \\ \text{subject to} & : & \boldsymbol{x} \in \mathcal{K} & \text{subject to} & : & \boldsymbol{z} \in \mathcal{K} \\ & & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} & & & \boldsymbol{A}^* \boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c} \end{array}$$

今回の SOCP $(\mathcal{P}_o)$,$(\mathcal{P}_p)$,$(\mathcal{P}_r)$ をソルバーに入力するするためには、上記の形に変形する必要がある。まず、目的関数に関しては内積以外を 0 に設定すれば良い。制約は、二次錐制約に関しては $z_i^q \in \mathcal{K}_q^{q_i}$、不等式制約に関しては $z^l \in \mathcal{K}_l^{n_l}$ を用いる。

例えば、$(\mathcal{P}_o)$ を入力するためには、双対問題を対象として以下のように設定をする。まず、変数は全て SDPT3 の $\boldsymbol{y}$ で扱うこととし、$\boldsymbol{y} = \boldsymbol{x}$ とする。次に、不等式制約は二次錐制約として対応させる。具体的には、不等式制約 $\boldsymbol{c}_i^T \boldsymbol{x} + d_i \geq \|\boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i\|$ は

$$\begin{pmatrix} -\boldsymbol{c}_i^T \\ -\boldsymbol{A}_i \end{pmatrix} \boldsymbol{x} + \boldsymbol{z}_i^q = \begin{pmatrix} d_i \\ \boldsymbol{b}_i \end{pmatrix}, \boldsymbol{z}_i^q \in \mathcal{K}_q^{q_i} \ (i = 1, \ldots, m)$$

の形に変形することで、$(A_i^q)^T y + z_i^q = c_i^q, z_i^q \in \mathcal{K}_q^{q_i} (i = 1, ..., n_q)$ の二次錐制約に対応させる。また、等式制約 $\boldsymbol{F}\boldsymbol{x} = \boldsymbol{g}$ は $A^u = \boldsymbol{F}^T, c^u = \boldsymbol{g}_u$ として

$$(A^u)^T y = c^u$$

の制約に対応させる。同様の手順により $(\mathcal{P}_p)$,$(\mathcal{P}_r)$ も SDPT3 に入力可能である。

5.2 初期点とパラメータの設定

SDPT3 は以下のような引数をとる。

$$[\text{obj,X,y,Z,info,runhist}] = \text{sqlp(blk,At,C,b,OPTIONS,X0,y0,Z0)}$$

ここで、blk は SOCP の錐を表すブロック構造のセル配列、At, C, b は SOCP の入力行列などを表す。また、OPTIONS は SDPT3 が用いるパラメータであり、X0 , y0 , Z0 は内点法の初期点を表す。

今回の数値実験では、OPTIONS,X0,y0,Z0 は SDPT3 が設定するデフォルト値を用いた。特に、デフォルトの場合の y0 の初期点は、次のように $(-1, 1)$ から発生する n 次元正規分布乱数ベクトルで設定される。

$$y0 = \text{randn}(n, 1)$$

得られた最適解の精度を評価するために、いくつかの定義を行う。まず、

$$\lambda_{\min,\mathcal{K}}(\boldsymbol{x}) = \min \left\{ \min_{j=1,\cdots,n_s} \lambda_{\min}(x_j^s), \min_{k=1,\cdots,n_q} \lambda_{\min,q}(x_k^q), \min_h(x_h^l) \right\}$$

とする。ここで、あるベクトル $\boldsymbol{a} = (a_1; \overline{\boldsymbol{a}})$ に対して $\lambda_{\min,q}(\boldsymbol{a}) = a_1 - \|\overline{\boldsymbol{a}}\|_2$, ある対称行列 $\boldsymbol{A}$ に対して $\lambda_{\min}(\boldsymbol{A}) = \boldsymbol{A}$ の最小の固有値とする。また、ベクトル $\boldsymbol{x} = (x_1, \cdots, x_n), \boldsymbol{z} = (z_1, \cdots, z_n) \in \mathbb{R}^n$ に対して、$\|\boldsymbol{x}\|_1 = \max_{i=1,\cdots,n} |x_i|$, $\langle \boldsymbol{c}, \boldsymbol{x} \rangle = \boldsymbol{c}^T \boldsymbol{x}$ とする。これらを踏まえて、以下の 6 つを定義する。これらは DIMACS error [**?**] とも呼ばれる。

$$\text{error1} = \frac{\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|}{1 + \|\boldsymbol{b}\|_1} \qquad \text{error2} = \max\{0, -\frac{\lambda_{\min,\mathcal{K}}(\boldsymbol{x})}{1 + \|\boldsymbol{b}\|_1}\}$$

$$\text{error3} = \frac{\|\boldsymbol{A}^* \boldsymbol{y} + \boldsymbol{z} - \boldsymbol{c}\|}{1 + \|\boldsymbol{c}\|_1} \qquad \text{error4} = \max\{0, -\frac{\lambda_{\min,\mathcal{K}}(\boldsymbol{z})}{1 + \|\boldsymbol{b}\|_1}\}$$

$$\text{error5} = \frac{\langle \boldsymbol{c}, \boldsymbol{x} \rangle - \boldsymbol{b}^T \boldsymbol{y}}{1 + \langle \boldsymbol{c}, \boldsymbol{x} \rangle + \boldsymbol{b}^T \boldsymbol{y}} \qquad \text{error6} = \frac{\langle \boldsymbol{x}, \boldsymbol{z} \rangle}{1 + \langle \boldsymbol{c}, \boldsymbol{x} \rangle + \boldsymbol{b}^T \boldsymbol{y}}$$

これらの Dimacs error には、以下の性質があることが知られている。

- error2 と error4 は、得られた解のうち $\boldsymbol{x}, \boldsymbol{z} \in \mathcal{K}$ であるならば、ともに 0 をとる。よって、得られた解の中で $\boldsymbol{x}, \boldsymbol{z}$ が二次錐制約を満たしているかどうかは、error2 と error4 の数値で確認することができる。
- 得られた解のうち $\boldsymbol{x}, \boldsymbol{z} \in \mathcal{K}$ が実行可能な場合、厳密な計算式では error5 = error6 となる。しかし、ソルバーで出力される解は数値誤差を含むので、必ずしも一致するとは限らない。
- error5 は双対ギャップを表しており、出力された解が厳密に実行可能解であれば弱双対定理より 0 以上である。しかし、数値誤差の影響により、error5 は負の値となる可能性がある。

これらを用いると、閾値を $\epsilon > 0$ としたときの SDPT3 の終了条件は以下のように示すことができる。

$$\text{終了条件：} \quad \max\{\text{error1}, \text{error2}, \ldots, \text{error6}\} < \epsilon$$

この条件を満たすまで、探索方向による計算を続け、条件を満たした時点で、その点を最適解として出力し、反復を終了する。なお、SDPT3 では、デフォルトの閾値は $\epsilon = 10^{-8}$ である。

5.3 テスト問題の生成方法

生成する問題 $(\mathcal{P}_o)$ については非有界であると数値誤差が非有界に起因するか内点がないことに起因するか判別が困難であるため、どの問題でも有界制約を追加し、必ず有界な問題となるように設定した。これにより、必ず最適解が存在するように問題を生成した。

具体的には、まず $i = 1, \ldots, m-1$ に対して $\boldsymbol{A}_i \in \mathbb{R}^{n \times n_i}, \boldsymbol{b}_i \in \mathbb{R}^n, \boldsymbol{c}_i \in \mathbb{R}^{n_i}$ を正規分布からなる乱数を要素に持つ変数として設定する。MATLAB では以下のように設定する。$\overline{x} = \text{randn}(n, 1) \in \mathbb{R}^n$ に対しては

$$A\{i\} = \text{randn}(n\_i, n), b\{i\} = \text{randn}(n\_i, 1), c\{i\} = \text{randn}(n, 1),$$
$$d\{i\} = -c\{i\}'\overline{x} + \text{norm}(A\{i\}'\overline{x} + b\{i\}) + 10$$

とする。このようにすることで、$i = 1, \ldots, m-1$ の場合の不等式制約を満たす変数 $\bar{x}$ が必ず存在することになる。次に、$\boldsymbol{A}_m \in \mathbb{R}^{n \times n_i}, \boldsymbol{b}_m \in \mathbb{R}^n, \boldsymbol{c}_m \in \mathbb{R}^{n_i}$ と、ある $\overline{\boldsymbol{x}} \in \mathbb{R}^{n_i}$ が満たすべき不等式制約 $\boldsymbol{c}_i^T \overline{\boldsymbol{x}} + d_i \geq \|\boldsymbol{A}_i \overline{\boldsymbol{x}} + \boldsymbol{b}_i\|$ に対して、$\boldsymbol{A}_m = \boldsymbol{I}_m, \boldsymbol{b}_m = \boldsymbol{0}, \boldsymbol{c}_i = \boldsymbol{0}$ とすることで、この不等式は

$$\boldsymbol{0} + d_m \geq \|\boldsymbol{I}_m \boldsymbol{x} + \boldsymbol{0}\| = \|\boldsymbol{x}\|$$

となる。よって、$d_m = \|\overline{\boldsymbol{x}}\| + 1$ と設定すれば、$i = m$ の場合の不等式制約は $\|\overline{\boldsymbol{x}}\| + 1 \geq \|\boldsymbol{x}\|$ となるので、有界な問題となる。この場合、MATLAB では $\bar{x} = \mathrm{randn}(n, 1) \in \mathbb{R}^n$ に対して以下のように設定する。

$$A\{m\} = \mathrm{eye}(n), b\{m\} = \mathrm{zeros}(n, 1), c\{m\} = \mathrm{zeros}(n, 1), d\{m\} = \mathrm{norm}(\bar{x})^2 + 1$$

5.4 実験結果

実験結果について、まずは計算時間と精度で評価をし、続いて反復回数による評価を行う。また、ペナルティ係数の設定が与える影響についても、検討する。

5.4.1 計算時間と精度の評価

この節では、3 手法による計算時間と Dimacs error 値の違いに着目をする。表 2 は、問題のサイズを変更した場合の各手法で求解までにかかった計算時間と Dimacs error の数値を表している。問題のサイズを決定する $m, n_i, \overline{m}$ の 3 つを変えて数値実験を行った。なお、ペナルティ法のペナルティ係数は 10^2 とした。

表 2 では、各ブロックにおいて横の行が各手法に対応しており、縦の列がそれぞれ SDPT3 に入力するように変換した後の変数の数、制約の数、計算時間 (秒)、Dimacs error に対応している。

表 2 から分かるように、計算時間は、通常の内点法とペナルティ法ではどのサイズの問題でもほぼ同じ計算時間であるが、緩和法は問題の規模が大きくなるとかなり長い計算時間を要する。これは、第 3.3 節で述べたように、生成される変数と制約の数が関係していることが分かる。

次に、error 値に着目する。まず、error2 と error4 について検討する。どの実験結果においても error2 と error4 が 0 に近い値となっており、全ての手法で実行可能解を求められていることが分かる。

また、等式条件と Dimacs error の対応について考えてみる。error1 は、主問題における等式制約に関する数値誤差、error3 は双対問題における等式制約に関する数値誤差、つまりどれくらい等式制約から逸脱しているかを表現している。error1 と error3 を見てみると、問題のサイズが小さい場合は、通常の内点法が 10^{-12} の精度で求解できているのに比べて、ペナルティ法と緩和法は 10^{-14} という精度で求解を達成できていることがわかる。次に、問題のサイズが大きい場合に注目すると、通常の内点法の精度は 10^{-12} と変わらないままとなっている。しかし、ペナルティ法は通常の内点法に比べて 10^{-7} まで精度が落ちてしまっているのに対して、緩和法はサイズが大きくなっても 10^{-13} という良い精度を保ちつつ求解ができていることがわかる。

表 2　各手法での計算時間と Dimacs error の数値

$m=3, n_i=5, \overline{m}=3$									
	変数の数	制約の数	計算時間 (秒)	error1	error2	error3	error4	error5	error6
通常の内点法	4	10	0.9	1.44E-11	0.00E+0	3.33E-11	0.00E+0	5.96E-09	6.58E-09
ペナルティ法	10	12	1.07	4.21E-16	0.00E+0	1.58E-11	0.00E+0	5.95E-09	7.57E-09
緩和法	39	16	1.37	9.14E-15	0.00E+0	3.18E-11	0.00E+0	7.54E-09	9.40E-09
$m=10, n_i=10, \overline{m}=10$									
	変数の数	制約の数	計算時間 (秒)	error1	error2	error3	error4	error5	error6
通常の内点法	4	31	1.14	1.19E-12	0.00E+0	1.18E-11	0.00E+0	8.53E-09	8.94E-09
ペナルティ法	24	33	1.24	4.86E-15	0.00E+0	3.93E-12	0.00E+0	4.72E-09	5.70E-09
緩和法	218	51	2.69	2.55E-14	0.00E+0	1.23E-11	0.00E+0	5.56E-09	7.60E-09
$m=30, n_i=10, \overline{m}=20$									
	変数の数	制約の数	計算時間 (秒)	error1	error2	error3	error4	error5	error6
通常の内点法	4	71	2.35	5.22E-13	0.00E+0	6.90E-12	0.00E+0	7.07E-09	7.53E-09
ペナルティ法	54	73	3.23	1.73E-14	0.00E+0	3.18E-12	0.00E+0	-1.74E-09	6.79E-09
緩和法	548	121	6.39	5.60E-14	0.00E+0	6.93E-12	0.00E+0	6.34E-09	8.53E-09
$m=20, n_i=20, \overline{m}=40$									
	変数の数	制約の数	計算時間 (秒)	error1	error2	error3	error4	error5	error6
通常の内点法	4	101	3.19	6.54E-12	0.00E+0	8.07E-12	0.00E+0	8.57E-09	9.78E-09
ペナルティ法	64	103	3.96	4.07E-14	0.00E+0	1.49E-12	0.00E+0	4.45E-09	9.26E-09
緩和法	1248	161	9.16	5.95E-14	0.00E+0	2.07E-12	0.00E+0	4.84E-09	7.09E-09
$m=50, n_i=50, \overline{m}=60$									
	変数の数	制約の数	計算時間 (秒)	error1	error2	error3	error4	error5	error6
通常の内点法	4	171	4.59	2.40E-11	0.00E+0	1.67E-12	0.00E+0	7.65E-09	8.52E-09
ペナルティ法	114	173	4.10	1.29E-07	0.00E+0	1.10E-06	0.00E+0	-7.03E-02	3.05E-04
緩和法	5568	281	48.35	9.30E-13	0.00E+0	6.97E-13	0.00E+0	6.72E-09	9.22E-09
$m=100, n_i=100, \overline{m}=80$									
	変数の数	制約の数	計算時間 (秒)	error1	error2	error3	error4	error5	error6
通常の内点法	4	261	6.21	2.03E-10	0.00E+0	3.96E-13	0.00E+0	5.26E-09	5.96E-09
ペナルティ法	184	263	6.57	2.80E-07	0.00E+0	2.70E-07	0.00E+0	-3.42E-02	6.20E-04
緩和法	18088	441	375.38	7.75E-13	0.00E+0	2.69E-13	0.00E+0	6.32E-09	8.02E-09

さらに error5, error6 についても考察を与える。error5 は双対ギャップに対応している。error5 を見てみると、通常の内点法と緩和法は 10^{-9} 程度の精度が安定して得られている。しかし、ペナルティ法では $m=30, n_i=10, \overline{m}=20$ と $m=50, n_i=50, \overline{m}=60$ および $m=100, n_i=100, \overline{m}=80$ の場合に負の値となっている。定理 4 より、本来 error5 は 0 以上の値を取るはずである。数値誤差の影響もあり、ペナルティ法では良い解が得られているわけではないことがわかる。error6 においても、ペナルティ法では精度値が落ちる場合が確認できる。

5.4.2　反復回数の評価

次に、各手法の求解に要する反復回数で比較を行う。ここでは、ペナルティ係数を 10^3 に設定した。問題のサイズ $m, n_i, \overline{m}$ をそれぞれ 1〜100 の範囲でランダムな値として生成した。具体的には

$$m, n_i, \overline{m} = 1 + \mathrm{round}(100 * \mathrm{rand})$$

とした。その条件の下で、各手法の反復回数を集計しグラフにしたものが図 1 である。合計 50 回実験を行い、横軸が実験番号を表し、縦軸が 3 手法の反復回数である。

平均回数は、通常の内点法が 24.6 回、ペナルティ法が 22.2 回、緩和法が 36.6 回となり、どの問題においても通常の内点法とペナルティ法に比べて、緩和法では最適解を求めるのに多い反復回数を必要とする。これは、緩和法では決定変数 $\boldsymbol{y}$ の数がかなり大きくなるため、他の 2 手法に比べて error 値の数値誤差が大きく出てしまうため、全ての error 値が ϵ 以下となるという反復の終了条件を満たしにくいことが影響していると考えられる。また、大体の場合において若干ペナルティ法

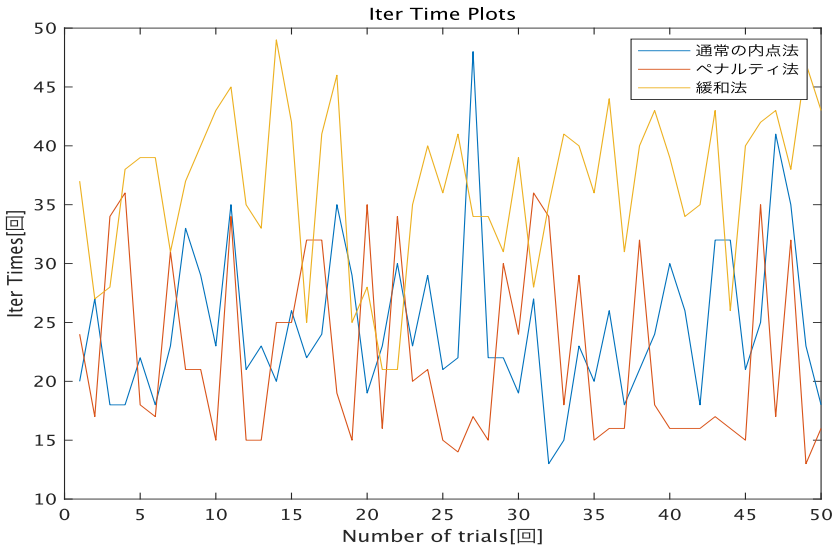

図 1　各手法の求解に要する反復回数

の方が通常の内点法に比べて少ない反復回数で求解できていることがわかる。これは、ペナルティ法では実行可能領域が広がるため、通常の内点法に比べてステップ長を少しだけ長く取ることができることが影響していると考えられる。

5.4.3　yet

ここでは、ペナルティ係数を 10^1〜10^5 の値からランダムに設定することで、ペナルティ係数の大きさが求解にどのような影響があるのかを調べる。5.4.2 節と同様に $i = 1, \cdots, m$ に対して

$$P_i, \overline{P}_i = 100 + \mathrm{round}(9900 * \mathrm{rand})$$

で生成した。計算時間と反復回数をまとめたのが図 2 のグラフである。

図 2 から、以下がわかる。求解にかかる反復回数は、ペナルティ係数を変化させた場合でも全て同じ 28 回となった。また、かかる CPU 計算時間は、概ね 7 秒と変化がなく安定していた。このことから、ペナルティ係数を変化させたとしても、反復回数と計算時間などに大きな影響はないことがわかった。より発展的な実験として、目的関数に追加するペナルティ係数を、二次関数や対数障壁関数などの異なる関数として追加した場合を考えると、また違った結果が得られるのではないかと考えられる。

次にペナルティ係数を変化させた場合の Dimacs error への影響について考察する。表 3 は、$(\mathcal{P}_p)$ においてペナルティ係数を変化させた場合の Dimacs error の数値を表している。ペナルティ係数の具体的な値としては、先ほどと同様 10^1〜10^5 の範囲の中で値を変化させて実験を行った。

表 3 からわかるように、error2 と error4 がともに 0 になっていたので、全ての場合で出力された解は二次錐に含まれている。$P, \overline{P}$ の値に着目すると、$P, \overline{P}$ の値が 10^2 以下の場合では、どの error 値も全て 0 に近い値となっていることから、安定して解を求めることができている。逆に、$P, \overline{P}$ の値が 10^5 のようにかなり大きい場合は、双対ギャップに対応する error5 が負の値となって

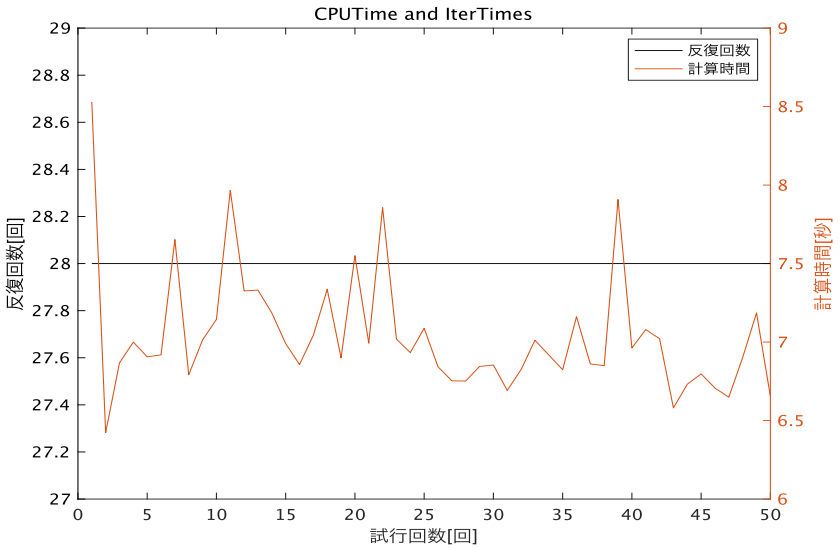

図 2　CPU 計算時間と反復回数

表 3　ペナルティ係数を変化させた場合の Dimacs error の数値

$m = 50, n_i = 50, \overline{m} = 50$						
	error1	error2	error3	error4	error5	error6
$P, \overline{P} = 10$	1.39E-13	0.00E+0	6.45E-14	0.00E+0	4.93E-09	5.58E-09
$P, \overline{P} = 10^2$	6.56E-13	0.00E+0	4.03E-14	0.00E+0	4.93E-09	5.63E-09
$P, \overline{P} = 10^3$	9.55E-14	0.00E+0	5.25E-14	0.00E+0	-1.40E-08	6.27E-09
$P, \overline{P} = 10^4$	1.57E-13	0.00E+0	3.92E-14	0.00E+0	-2.82E-09	5.41E-09
$P, \overline{P} = 10^5$	9.59E-11	0.00E+0	3.27E-07	0.00E+0	-0.999	1.47E-05

いることからうまく求解できていないことがわかる。$P, \overline{P}$ の値が $10^3, 10^4$ の場合は、error5 が負の値となっている。しかし、この数値実験では多少の数値誤差を含むので、error5 がほぼ 0 に近い値となっていること、および、その他の error 値が $P, \overline{P} = 10, 10^2$ の時と同じ精度で正の値かつ 0 に近い値となっていることから、安定して求解できていると考えられる。

5.4.4　yet

3.2 節で見たように、ペナルティ法では不等式制約に $\xi_i, \overline{\xi}_i$ を加えて実行可能領域を広げ、必ず実行可能内点を持つようにすることで数値的安定性の向上を目指した。ここで、追加した変数 $\xi_i, \overline{\xi}_i$ に着目し、追加する前の問題 $(\mathcal{P}_o)$ と追加した後の問題 $(\mathcal{P}_p)$ で、どれくらい実行可能領域が変化しているかを考える。

そのために、$\xi_i, \overline{\xi}_i$ の値に着目する。今回は、$m = 10, \overline{m} = 10$ として数値実験を行った。その他は、$n_i = 10, P = 10^2$ と設定した。以前の実験と同様に有界性も仮定した。得られた具体的な

$\xi_i, \overline{\xi}_i$ の値は以下である。

$$\xi = \begin{pmatrix} 2.9638E-12 \\ 2.9665E-12 \\ 2.9633E-12 \\ 2.9655E-12 \\ 2.9687E-12 \\ 2.9638E-12 \\ 2.9643E-12 \\ 2.9649E-12 \\ 2.9666E-12 \\ 2.9597E-12 \end{pmatrix}, \ \overline{\xi} = \begin{pmatrix} 3.7917E-12 \\ 3.7649E-12 \\ 3.8559E-12 \\ 3.5737E-12 \\ 3.6003E-12 \\ 3.5165E-12 \\ 3.8981E-12 \\ 4.0037E-12 \\ 3.7935E-12 \\ 4.0776E-12 \end{pmatrix}$$

この数値から、各ベクトル ξ と $\overline{\xi}$ の各成分にばらつきはなく、ほぼ同じ値となる結果となった。また、若干 $\overline{\xi}$ の成分の方が ξ の成分よりも大きい値となった。具体的な値としては、10^{-12} の精度となっていてほぼ 0 に近い値である。このことから、最適解の領域はあまり広がらず、元の問題 $(\mathcal{P}_o)$ と同じ制約を保ったまま求解できていることがわかる。

6　結論

本論文では、実行可能内点を持たないような SOCP を内点法で求解する場合に不安定になりやすいことを解消する方法として、緩和法とペナルティ法を施すことで、数値的安定性の向上を目指した。第 5 節の結果から、求解の精度は、どの問題のサイズに対しても緩和法が他の手法に比べて有効であることがわかった。逆に、求解にかかる反復回数は、緩和法が他の手法に比べて少し多い回数を必要とする結果となった。計算時間は、緩和法では変数の数が多くなるため、通常の内点法とペナルティ法に比べて、求解までに要する計算時間が長くなることが分かった。また、ペナルティ法のみに着目してペナルティ係数を変化させた場合では、ペナルティ係数を 10^4 以上の値だと安定して求解できない結果となったが、それより小さい値で設定をすることで良い精度を保ったまま安定して解を求めることができていることが確認された。求解にかかる反復回数は問題のサイズを変えても同じ回数を要する結果となった。

今後の課題としては、以下が挙げられる。ペナルティ法による変形について、目的関数に追加するペナルティ項を 1 次関数で追加したので、二次関数などの別の関数を使ったペナルティ項を追加して実験をすることが考えられる。緩和法については、問題のサイズを大きくした際、求解にかかる時間が急激に増えてしまうというデメリットがわかった。緩和法の定式化に基づく効率的な内点法を設計するなどして、求解にかかる時間を短くすることも今後の課題と考えられる。

謝辞

本論文の執筆にあたり、数多くの助言や論文の添削、さらには数多くの質問に対して優しく丁寧に指導して頂きました指導教員の山下真教授に感謝いたします。また、ゼミを通じて多くのことを教えてくださった山下研究室、澄田研究室の先生方や先輩方に感謝いたします。最後に、学士論文をともに取り組んだ山下研究室、澄田研究室の仲間や友人、支えてくださった家族には感謝いたします。

参考文献