ReSNAの手引き

林　俊介

与えられた関数に対して2次錐相補性条件（2次錐上で定義された相補性条件）や等式条件を満たすようなベクトルを求める問題を2次錐相補性問題という．本稿では，2次錐相補性問題を解くために開発された，正則化平滑化ニュートン法に基づくソルバーであるReSNA (Regularized Smoothing Newton Algorithm) について，その使い方や背景を紹介する．なお，ReSNAは2次錐相補性問題だけでなく，非線形2次錐計画問題や通常の（非負象限で定義される）非線形相補性問題，混合相補性問題にも対応している．特に，ReSNAプラグインを用いることにより，解きたい問題と2次錐相補性問題との関係を知らなくても，直接的にReSNAを適用することができる．また，本稿では，ReSNAの用法だけでなく，アルゴリズムの理論的背景や収束性についても述べる．

キーワード：2次錐相補性問題，2次錐計画問題，相補性問題，正則化平滑化ニュートン法，Matlab

1. はじめに

ReSNAとは，筆者がウェブサイト[1]上で公開している2次錐相補性問題[1]を解くためのMatlab用ソフトウェアである．ソフトウェアとは言っても，もともと，論文の数値実験用のMatlabファイルだったものに少し手を加えただけに過ぎない．したがって，ソフトウェアを名乗るのは大変お恥ずかしいところだが，Wikipedia [7]によると「コンピューターシステム上で何らかの処理を行うプログラムや手続き，およびそれらに関する文書を指す言葉」であれば，ソフトウェアと名乗ってもよいらしいので，ご容赦いただければ幸いである．

ところで，ReSNAの核となるアルゴリズムは，正則化平滑化ニュートン法(Regularized Smoothing Newton Algorithm)であり，ReSNAの名称もその頭文字に由来する．アルゴリズム自体は，約10年前に筆者が当時の指導教官であった福島雅夫先生，山下信雄先生と共に開発したものであり，その成果は論文[8]に詳細にまとめられている．当時の筆者はまだまだ未熟な学生であり，論文に載せる数値実験のためだけにMatlabのファイルを作成したため，人様に使っていただくという発想もなく，プログラムコードもただただ乱雑に書かれていた．しかしながら，その後，当時リスボン工科大学に在籍されていた寒野善博先生（現・東京大学）から「建築に現れるある種の問題が2次錐相補性問題として定式化され[9]，それを数値的に解いてみたいので，2次錐相補性問題のソルバーをもらえないか」という内容のご依頼をいただき，そのとき初めて自作のアルゴリズムを他人に使ってもらう経験をした次第である．その際，アフィン関数のみを含む線形2次錐相補性問題のソルバーを整備し，それをウェブサイトの片隅にこっそりとアップロードしたのだが，敢えて名前を付けたりマニュアルを整備したりすることもなく，研究室の学生の卒論や修論の数値実験用として内輪で使ってきた．その後，2013年に筆者が東北大に異動し，同僚の先生から「非線形な相補性問題を解くための適当なソルバーがないか？」という話をいただいた際に，2次錐やジョルダン代数などを知らなくても気軽に実装できるプログラムを作ってみようと思い立ち，マニュアルを整備してReSNAと名付けてウェブにアップロードした．以上がこれまでの経緯である．

なお，ReSNAの特徴としてプラグインを同梱していることが挙げられる．ReSNAに用いられている正則化平滑化ニュートン法は，2次錐相補性問題だけでなく，非線形2次錐計画問題や非線形相補性問題などにも適用することができるが，そのためには2次錐計画問題のKarush–Kuhn–Tucker (KKT)条件や2次錐と非負錐の関係などを知っておかなければならない．これらのことは，連続最適化の研究者であれば大抵知っていることだが，そうでない方々にとっては意外に馴染みが薄い．そこで，非線形相補性問題ならこのプラグイン，2次錐計画問題ならこのプラグインといった

[1] 2次錐上で定義される相補性問題のこと．応用としてロバストNash均衡問題[2, 3]やロバストWordrop均衡問題[4, 5]などが挙げられる．詳しくは本稿2節ないし[6]を参照のこと．

はやし　しゅんすけ
東北大学情報科学研究科
〒980–8579 宮城県仙台市青葉区荒巻字青葉6–3–09

感じで，ユーザーが興味のあるクラスの問題だけに特化して使えるよう，いくつかの代表的なクラスの問題に対してプラグインを用意している．ただし，プラグインと言っても，ただ単に解きたい問題をReSNA本体の記述形式に変換するだけのファンクションM-ファイルに過ぎず，計算自体はReSNA本体が担当することになる．

ところで，2次錐相補性問題は線形2次錐計画問題[2]をサブクラスとして含むため，ReSNAはこれらの問題にも適用可能である．しかしながら，これらの問題を解きたいのであれば，ReSNAよりもSeDuMi, SDPT3，SDPA [10〜12] の方を強くお勧めする．なぜなら，ReSNAには行列演算などの数値計算テクニックが組み込まれていないため，大規模な2次錐計画問題を解く際には，無駄な計算コストがかかってしまうからである．他にも，ユーザーインターフェースが整備されていなかったり，バージョンアップの予定がなかったりと，様々な面でReSNAは『ソフトウェア』と名乗るには不十分であることは否めない．しかし，理論的には優れた収束性[3]が保障されているし，非線形2次錐計画問題や各種の相補性問題に対して気軽に適用できるということもあるので，比較実験のネタに用いるなど，それなりの存在価値はあるはずだと信じたいところである．

2. 2次錐相補性問題とその関連問題

自然数sが与えられたとき，以下のように定義される閉凸錐$\mathcal{K}^s \subset \mathbb{R}^s$を$s$次元の2次錐 (Second-Order Cone: SOC) という：

$$\mathcal{K}^s := \begin{cases} \{x \in \mathbb{R} \mid x \geq 0\} & (s=1) \\ \left\{(x_1, \tilde{x}) \in \mathbb{R} \times \mathbb{R}^{s-1} \;\middle|\; x_1 \geq \|\tilde{x}\|_2\right\} & (s \geq 2). \end{cases}$$

2つのベクトルx，yがいずれも同じ2次錐もしくはその直積に含まれ，互いに直交しているとき，xとyは2次錐相補性条件を満たすという．ReSNAで対象とするのは，次のような等式条件と2次錐相補性条件の含まれた混合型の2次錐相補性問題 (Second-Order Cone Complementarity Problem: SOCCP) である：

$$\begin{array}{rl} \text{Find} & (x,y,p) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^l \\ \text{such that} & x \in \mathcal{K},\ y \in \mathcal{K},\ x^\top y = 0, \\ & y = F(x,p),\ G(x,p) = 0. \end{array} \tag{1}$$

ここで，$F: \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^n$，$G: \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^l$は与えられた関数であり，$\mathcal{K} \subset \mathbb{R}^n$は$n_i$次元の2次錐を用いて

$$\mathcal{K} = \mathcal{K}^{n_1} \times \mathcal{K}^{n_2} \times \cdots \times \mathcal{K}^{n_m} \tag{2}$$

（ただし，$n = n_1 + n_2 + \cdots + n_m$）で定義される閉凸錐である．

前述の定義より，n次元の非負象限$\mathbb{R}^n_+ := \{x \in \mathbb{R}^n \mid x \geq 0\}$は，$\mathcal{K}^1 \times \cdots \times \mathcal{K}^1$と書くことができる．したがって，混合相補性問題 (Mixed Complementarity Problem: MCP)

$$\begin{array}{rl} \text{Find} & (x,y,p) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^l \\ \text{such that} & x \geq 0,\ y \geq 0,\ x^\top y = 0, \\ & y = F(x,p),\ G(x,p) = 0 \end{array}$$

はSOCCP (1) において$\mathcal{K} := \mathcal{K}^1 \times \cdots \times \mathcal{K}^1$としたものにほかならない．さらに，非線形相補性問題 (Nonlinear Complementarity Problem: NCP)

$$\begin{array}{rl} \text{Find} & (x,y) \in \mathbb{R}^n \times \mathbb{R}^n \\ \text{such that} & x \geq 0,\ y \geq 0,\ x^\top y = 0,\ y = F(x) \end{array} \tag{3}$$

はSOCCP (1) において$\mathcal{K} := \mathcal{K}^1 \times \cdots \times \mathcal{K}^1$，$l = 0$，$p \equiv \emptyset$，$G \equiv \emptyset$としたものであり，さらに関数$F$がアフィン関数ならば，SOCCP (1) は線形相補性問題 (Linear Complementarity Problem: LCP) に帰着される．一方，非線形2次錐計画問題 (Nonlinear Second-Order Cone Program: NSOCP) は，関数$\theta: \mathbb{R}^t \to \mathbb{R}$，$g: \mathbb{R}^t \to \mathbb{R}^n$，$h: \mathbb{R}^t \to \mathbb{R}^k$を用いて

$$\begin{array}{rl} \text{minimize} & \theta(z) \\ \text{subject to} & g(z) \in \mathcal{K},\ h(z) = 0 \end{array} \tag{4}$$

と表される問題であるが，この問題のKarush–Kuhn–Tucker (KKT) 条件は

$$\begin{array}{l} \nabla\theta(z) - \nabla g(z)x - \nabla h(z)w = 0 \\ x \in \mathcal{K},\ g(z) \in \mathcal{K},\ x^\top g(z) = 0,\ h(z) = 0 \end{array} \tag{5}$$

と書くことができる[4]．よって，KKT条件 (5) において

[2] アフィン関数のみで記述される2次錐計画問題のこと．これをもって単に2次錐計画問題と呼ぶことも多い．

[3] 非混合型の2次錐相補性問題に対して，単調性の仮定の下で，大域的収束性と2次収束性が証明されている．詳しくは本稿5節または [8] を参照のこと．

[4] $g: \mathbb{R}^t \to \mathbb{R}^n$に対して$\nabla g(z) := (\nabla g_1(z), \ldots, \nabla g_n(z)) \in \mathbb{R}^{t \times n}$である．これはヤコビアンの転置行列をとったものに相当するので，転置ヤコビアンとも呼ばれる．

$$p := \begin{pmatrix} z \\ w \end{pmatrix},\ l := t + k,\ F(x,p) := g(z),$$
$$G(x,p) := \begin{pmatrix} \nabla\theta(z) - \nabla g(z)x - \nabla h(z)w \\ h(z) \end{pmatrix}$$

とすれば，SOCCP (1) として等価に表すことができる．すなわち，2 次錐計画問題は適当な制約想定の下で 2 次錐相補性問題として解くことができる．

3. ReSNA の用法と実行例

本節では，ReSNA の用法について具体的に述べるとともに，実行例として，簡単なテスト問題を解いた結果をいくつか紹介する．なお，タイプライターフォントで書かれたものは，Matlab のファイル名，コマンド，変数などを表す．

3.1 メインプログラムの入出力

ReSNA の核となるファンクション M-ファイルは `ReSNA.m` である．このプログラムは非線形等式混合型の SOCCP (1) に対応しており，具体的な用法は

```
[x,y,p] = ReSNA(FUNC,nabFUNC,K,el,x0,y0,p0)
```

である．それぞれの入力の意味は以下のとおりである．

- `FUNC` — 以下で定義される関数 $\Gamma : \mathbb{R}^{n+l} \to \mathbb{R}^{n+l}$ を表す．

$$\Gamma(z) = \begin{pmatrix} F(x,p) \\ G(x,p) \end{pmatrix},\ z = \begin{pmatrix} x \\ p \end{pmatrix}. \tag{6}$$

  たとえば，関数 Γ をファンクション M-ファイル `Gm.m` で定義したならば，`Gm.m` を `ReSNA.m` と同じフォルダに置き，`FUNC = @Gm` とする．なお，`Gm.m` の入力と出力は，いずれも $n + l$ 次元の列ベクトル 1 つのみでなければならない．後述の入力 `el` によって，入力 z を x と p に，出力 Γ を F と G に自動的に分解する．

- `nabFUNC` — $\nabla\Gamma : \mathbb{R}^{n+l} \to \mathbb{R}^{(n+l)\times(n+l)}$，すなわち，上記の関数 Γ の導関数（転置ヤコビアン）

$$\nabla\Gamma(z) = \begin{pmatrix} \nabla_x F(x,p) & \nabla_x G(x,p) \\ \nabla_p F(x,p) & \nabla_p G(x,p) \end{pmatrix},\ z = \begin{pmatrix} x \\ p \end{pmatrix}$$

  を表す．たとえば，関数 $\nabla\Gamma$ をファンクション M-ファイル `nabGm.m` で定義したならば，`nabGm.m` を `ReSNA.m` と同じフォルダに置き，`nabFUNC = @nabGm` とする．なお，`nabGm.m` の入力は $n + l$ 次元の列ベクトル 1 つのみ，出力は $(n+l) \times (n+l)$ の正方行列 1 つのみでなければならない．また，関数値 $\nabla\Gamma(z)$ を陽に計算できない場合は，`nabFUNC = []` とすれば，$\nabla\Gamma(z)$ の値を有限差分近似で自動的に計算する．ただし，この場合は少なからぬ計算誤差が生じることに注意されたい．
- `K` — SOCCP (1) における 2 次錐の直積 $\mathcal{K}$ の次元構造を表す．具体的には，`K` はその成分が各 2 次錐の次元と一致するような列ベクトルもしくは行ベクトルとして与える．たとえば，(2) において $\mathcal{K} = \mathcal{K}^3 \times \mathcal{K}^1 \times \mathcal{K}^2$ ならば `K = [3,1,2]`，$\mathcal{K} = \mathbb{R}^{10}_{+}$ ならば `K = ones(1,10)` とする．
- `el` — l の値，すなわち SOCCP (1) における変数 p および関数値 $G(x,p)$ の次元を表す．当然，0 以上 $n + l$ 以下の整数でなければならない．もし p および $G(x,p)$ が存在しない（等式混合型でない）場合は `el = 0` とする．
- `x0` — アルゴリズムにおける初期点 $x^{(0)}$ を表す．よって，次元が `sum(K)` であるような列ベクトルでなければならない．なお，`x0 = []` とした場合，自動的に $[-1,1]^n$ の中からランダムに選択される．
- `y0` — アルゴリズムにおける初期点 $y^{(0)}$ を表す．よって，次元が `sum(K)` であるような列ベクトルでなければならない．なお，`y0 = []` とした場合，自動的に $[-1,1]^n$ の中からランダムに選択される．
- `p0` — アルゴリズムにおける初期点 $p^{(0)}$ を表す．よって，次元が `el` であるような列ベクトルでなければならない．なお，`p0 = []` とした場合，自動的に $[-1,1]^l$ の中からランダムに選択される．

なお，`x0,y0,p0` をすべて省略する場合，入力を `el` までとしても構わない．それから，出力 `x,y,p` はそれぞれ，SOCCP (1) の解 (x,y,p) を表す．

3.2 可変パラメータ

`ReSNA.m` の冒頭に記述されているパラメータのいくつかは，テキストエディタなどで変更することができる．それらの意味するところは以下のとおりである．

- `PROGRESS` — 各反復における関数値やパラメータ値などの詳細を Matlab のコマンドラインに表示する (`PROGRESS='Y'`) か表示しない (`PROGRESS='N'`) かを決定する．デフォルトは`'Y'`．
- `tole` — アルゴリズムの終了条件に用いる許容値．この値が 0 であれば，アルゴリズムは理論上の厳密解を出力するが，反復回数が無限大になってしまう可能性がある．後述のアルゴリズム 1 の Step 1 において，終了条件を $\|H_{\mathrm{NR}}(w^{(k)})\| \le$ `tole` とすることに対応する．デフォルト値は 10^{-8}．
- `tole_diff` — 導関数を差分近似で求める際に用

いる差分量の値．大きすぎても小さすぎても誤差が大きくなるので注意されたい．デフォルト値は 10^{-8}．

- eta, eta_bar, rho, sigma, kappa, kappa_bar, kappa_hat — 後述のアルゴリズム 1 に用いられるパラメータ η，$\overline{\eta}$，ρ，σ，κ，$\overline{\kappa}$，$\hat{\kappa}$ に対応する．デフォルト値は ReSNA.m の記載値を参照のこと．

3.3 ReSNA プラグイン

2 節で述べたように，MCP，NCP，LCP，NSOCP などは SOCCP (1) にサブクラスとして含まれる．よって，これらの問題は適当な変換を施すことにより ReSNA.m で解くことができる．しかし，これらの変換には少しばかりの知識と手間を要する．そこで，解きたいクラスの問題を ReSNA.m の入力様式に自動的に変換してくれるのが，ReSNA プラグインである．ここでは，そのいくつかを紹介する．

- MLSOCCP_ReSNA.m — このプラグインは，混合型の線形 2 次錐相補性問題

$$\begin{array}{rl} \text{Find} & (x,y,p)\in\mathbb{R}^n\times\mathbb{R}^n\times\mathbb{R}^l \\ \text{such that} & x\in\mathcal{K},\ y\in\mathcal{K},\ x^\top y=0, \\ & y=M_{11}x+M_{12}p+q_1, \\ & M_{21}x+M_{22}p+q_2=0 \end{array}$$

に対応したものである．ここで，$M_{11}\in\mathbb{R}^{n\times n}$，$M_{12}\in\mathbb{R}^{n\times l}$，$M_{21}\in\mathbb{R}^{l\times n}$，$M_{22}\in\mathbb{R}^{l\times l}$，$q_1\in\mathbb{R}^n$，$q_2\in\mathbb{R}^l$ は所与の行列およびベクトルである．このとき，MLSOCCP_ReSNA.m の入力は M,q,K,el,x0,y0,p0 であり，FUNC や nabFUNC のような関数を入力とする必要がない．ただし，M と q はそれぞれ以下のような行列とベクトルを表す．

$$M=\begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22}\end{pmatrix},\quad q=\begin{pmatrix} q_1 \\ q_2\end{pmatrix} \tag{7}$$

- NCP_ReSNA.m — このプラグインは，非線形相補性問題 (NCP) (3) に対応したものである．このとき，NCP_ReSNA.m の入力は FUNC,nabFUNC,n,x0,y0 である．ここで，FUNC,nabFUNC はそれぞれ関数 $F:\mathbb{R}^n\to\mathbb{R}^n$ および $\nabla F:\mathbb{R}^n\to\mathbb{R}^{n\times n}$ に対応する．また，n は決定変数 x，y および関数値 $F(x)$ の次元 n に対応する．
- NSOCP_ReSNA.m — このプラグインは，非線形 2 次錐計画問題 (NSOCP) (4) に対応したものである．（2014 年 9 月 5 日現在，本プラグインは完成していないが，近日中にウェブ上にアップロードする予定である．）

その他，ReSNA には 10 個以上のプラグインが用意されているが，どのプラグインを使えばよいかわからない場合は，まず@@Read_me_first!フォルダ内の manual_plugin.pdf をご参照いただきたい．

3.4 ナブラチェッカー

ReSNA 本体や非線形関数を扱うプラグインでは，非線形関数の導関数（転置ヤコビアン）を代数演算で記述して入力関数とする必要がある[5]が，それを手計算で行うと往々にして計算間違いを犯してしまう．そこで，導関数が正しく記述できているかどうかをチェックしてくれるのがナブラチェッカー (nabla_checker.m) である．仕組みは単純で，10 個のランダムな入力に対して，代数演算と差分演算による計算値の相対誤差がすべて閾値以下であるかどうかによって，代数演算が正しいかどうかを判断している．

3.5 実行例

次に，ReSNA の実行例として，具体的な問題に適用した結果をいくつか紹介する．

ReSNA にはテスト関数として，FF.m が同梱されている．これは，以下で定義される非線形関数 $\Gamma:\mathbb{R}^5\to\mathbb{R}^5$ を記述したものである：

$$\Gamma(z):=\frac{2Az}{1+\exp(-z^\top Az)}+b+0.01z, \tag{8}$$

$$A=\begin{bmatrix} 4.92 & -2.76 & -5.12 & 0.60 & 4.60 \\ -2.76 & 3.96 & 3.12 & 2.28 & -2.28 \\ -5.12 & 3.12 & 5.68 & -1.08 & -5.60 \\ 0.60 & 2.28 & -1.08 & 4.52 & 2.76 \\ 4.60 & -2.28 & -5.60 & 2.76 & 6.52 \end{bmatrix},$$

$$b=\begin{bmatrix} 0.09 & -0.41 & 0.49 & -0.62 & 0.37 \end{bmatrix}^\top.$$

このとき，導関数を代数演算で計算すると，

$$\nabla\Gamma(z)=\frac{2A}{1+\exp(-z^\top Az)} + \frac{(2Az)\exp(-z^\top Az)(2Az)^\top}{(1+\exp(-z^\top Az))^2}+0.01I$$

となり，これを記述したのが FFd.m である．実際，図 1 が示すように，ナブラチェッカーによっても上記の代数演算が正しい（可能性が非常に高い）ことが確認できる．ただし，nabla_checker.m の入力の 5,5 は，FF.m の入力ベクトルと出力ベクトルがそれぞれ 5 次元であることに対応している．

[5] 有限差分近似を用いれば導関数を入力する必要はないが，どうしても丸め誤差などが発生してしまうので，代数的に導関数の計算ができる場合は，それを計算して入力関数とするのが望ましい．

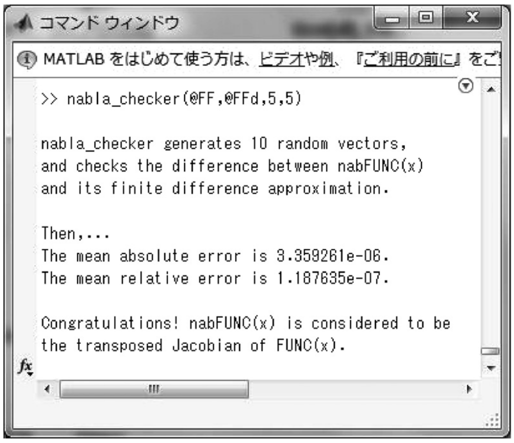

図 1 `nabla_checker.m` の実行例

図2は，(8) で定義された関数 Γ を含む SOCCP (1) に対して，`ReSNA.m` を適用した結果を示したものである．ただし，$n=4$，$l=1$，$\mathcal{K}=\mathcal{K}^3\times\mathbb{R}_+$ であり，関数 $F:\mathbb{R}^4\times\mathbb{R}\to\mathbb{R}^4$ および $G:\mathbb{R}^4\times\mathbb{R}\to\mathbb{R}$ は (6) で与えられている．ここで，`ReSNA.m` の入力 `@FF,@FFd,[3 1],1` はそれぞれ関数 Γ を表すファンクション M-ファイル `FF.m`，関数 $\nabla\Gamma$ を表すファンクション M-ファイル `FFd.m`，二次錐の直積構造 $\mathcal{K}=\mathcal{K}^3\times\mathbb{R}_+$，および変数 p の次元 $l=1$ に対応している．ここで，$3+1+1=$（`FF.m` の入出力ベクトルの次元）$=5$ であることに注意する．また，`(Iteration Information)` は，アルゴリズムの各反復における関数値やパラメータ値を出力したものである．具体的には，`k,j,m` はそれぞれ後述のアルゴリズム 1 における外部反復 k，内部反復 j，およびアルミホのステップサイズルールに用いる非負整数 m に対応している．また，`mu,epsi,beta` はそれぞれ μ_k，ε_k，β_k を，`|Hme|,|Hnr|` はそれぞれ $\|H_{\mu_k,\varepsilon_k}(w^{(k)})\|$，$\|H_{\mathrm{NR}}(w^{(k)})\|$ を表している．これらについて，詳細は本稿 5 節を参照のこと．

図 3 は同梱プラグインの 1 つである `MLCP_ReSNA.m` を線形な MCP:

$$\begin{aligned} \text{Find} \quad & (x,y,p)\in\mathbb{R}^3\times\mathbb{R}^3\times\mathbb{R}^2 \\ \text{such that} \quad & x\geq 0,\ y\geq 0,\ x^\top y=0, \\ & y=M_{11}x+M_{12}p+q_1, \\ & M_{21}x+M_{22}p+q_2=0 \end{aligned}$$

に対して適用した結果を示したものである．ただし，

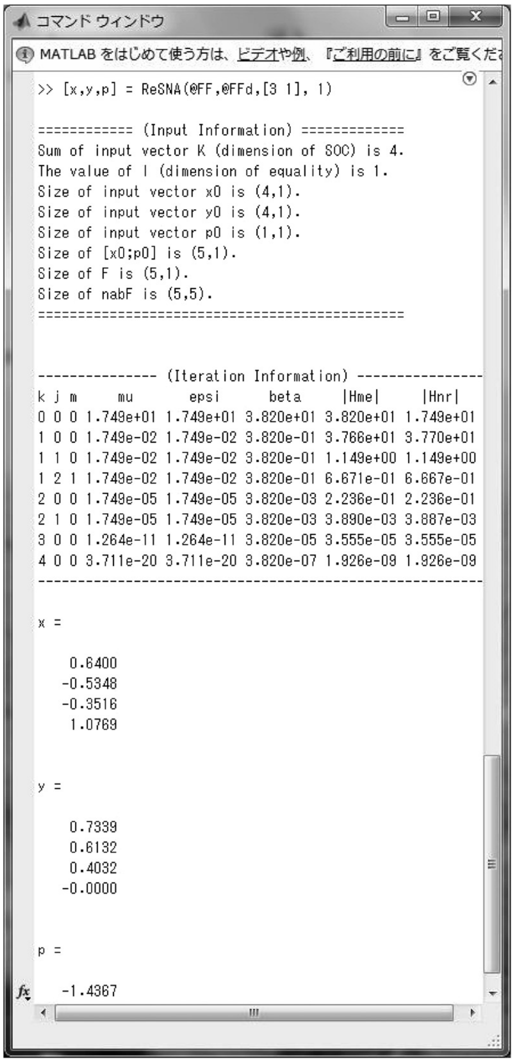

図 2 `ReSNA.m` の実行例

$$M_{11}=\begin{bmatrix}2.0 & -0.7 & -0.1\\ -0.7 & 2.3 & 0.7\\ -0.1 & 0.7 & 1.1\end{bmatrix},\ M_{12}=\begin{bmatrix}-0.5 & 1.0\\ 1.4 & -0.5\\ 0.1 & -0.5\end{bmatrix},$$

$$M_{21}=\begin{bmatrix}-0.5 & 1.4 & 0.1\\ 1.0 & -0.5 & -0.5\end{bmatrix},\ M_{22}=\begin{bmatrix}1.5 & 0.2\\ 0.2 & 1.2\end{bmatrix},$$

$$q_1=\begin{bmatrix}0.7 & -0.3 & 0.6\end{bmatrix}^\top,\ q_2=\begin{bmatrix}0.4 & -1.0\end{bmatrix}^\top$$

である．なお，`MLCP_ReSNA.m` の入力 `M,q,2` はそれぞれ，(7) で表される行列 $M\in\mathbb{R}^{5\times 5}$，ベクトル $q\in\mathbb{R}^5$，および変数 p の次元 2 に対応していることに注意する．

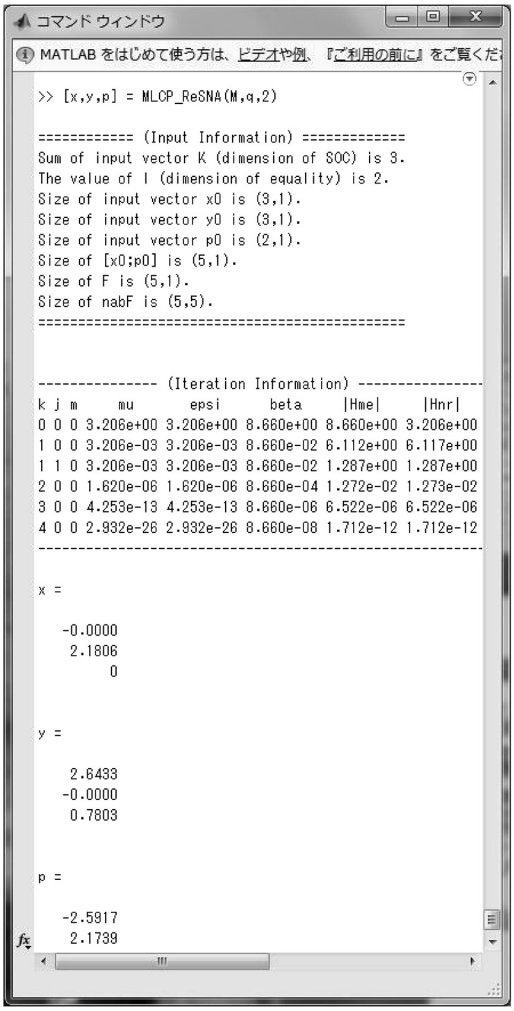

図 3 MLCP_ReSNA.m の実行例

4. 正則化平滑化ニュートン法の考え方

本節では，ReSNA で用いられている手法である正則化平滑化ニュートン法の考え方や理論的背景について述べる．

4.1 等価なベクトル方程式への変換

SOCCP (1) における決定変数 x, y を 2 次錐の直積構造 (2) に応じて $x = (x^1, \ldots, x^m) \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_m}$, $y = (y^1, \ldots, y^m) \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_m}$ と分解する．このとき，次のように定義される関数 $\Phi_{\mathrm{NR}} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ を残差関数という：

$$\Phi_{\mathrm{NR}}(x, y) := \begin{pmatrix} \varphi_{\mathrm{NR}}(x^1, y^1) \\ \vdots \\ \varphi_{\mathrm{NR}}(x^m, y^m) \end{pmatrix},$$

$$\varphi_{\mathrm{NR}}(x^i, y^i) := x^i - P_0(x^i - y^i),$$

$$P_0(z) := \max\{0, \lambda_1\} u^{\{1\}} + \max\{0, \lambda_2\} u^{\{2\}}.$$

ここで，λ_1 および λ_2 はベクトル z のスペクトル値，$u^{\{1\}}$ および $u^{\{2\}}$ は z のスペクトルベクトルを表す[6]．なお，関数 $P_0(z)$ は，ベクトル z の 2 次錐 $\mathcal{K}^{n_i}$ に対する直交射影と一致することが知られている．このとき，

$$\Phi_{\mathrm{NR}}(x, y) = 0 \iff x \in \mathcal{K},\ y \in \mathcal{K},\ x^T y = 0$$

が成り立つ [14, 15] ので，関数 $H_{\mathrm{NR}} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^{2n+l}$ を

$$H_{\mathrm{NR}}(x, y, p) := \begin{pmatrix} \Phi_{\mathrm{NR}}(x, y) \\ F(x, p) - y \\ G(x, p) \end{pmatrix} \tag{9}$$

で定義することにより，SOCCP (1) はベクトル方程式

$$H_{\mathrm{NR}}(x, y, p) = 0 \tag{10}$$

と等価になることがわかる．

以上より，SOCCP (1) を直接解く代わりにベクトル方程式 (10) を解けばよい．しかしながら，関数 Φ_{NR} は微分不可能であるため，ニュートン法を直接適用することができない．さらに，関数 F や G がよい性質を持っていたとしても，$\|H_{\mathrm{NR}}(x, y, p)\|$ のレベル集合が有界とは限らず，大域的収束性の理論的保障が期待できない．そこで，これらの弱点をカバーするため，平滑化法と正則化法を導入する．

4.2 平滑化法

平滑化法では，微分不可能な関数の代わりに平滑化パラメータによって微分可能に近似された関数を用いる．ここでは，前節で出てきた残差関数 Φ_{NR} の代わりに平滑化パラメータ $\mu > 0$ で制御される関数 Φ_μ を用いる．その具体的な定義は以下のとおりである：

$$\Phi_\mu(x, y) := \begin{pmatrix} \varphi_\mu(x^1, y^1) \\ \vdots \\ \varphi_\mu(x^m, y^m) \end{pmatrix},$$

$$\varphi_\mu(x^i, y^i) := x^i - P_\mu(x^i - y^i),$$

$$P_\mu(z) := \gamma_\mu(\lambda_1) u^{\{1\}} + \gamma_\mu(\lambda_2) u^{\{2\}}.$$

[6] スペクトル分解（固有値分解）の詳細は [6, 13, 14] を参照のこと．

ここで，λ_1 および λ_2 はベクトル z のスペクトル値，$u^{\{1\}}$ および $u^{\{2\}}$ は z のスペクトルベクトルを表す．また γ_μ は，3 つの条件 (i) $\lim_{\alpha\to-\infty}\hat{g}(\alpha)=0$, (ii) $\lim_{\alpha\to\infty}(\hat{g}(\alpha)-\alpha)=0$, (iii) $0<\hat{g}'(\alpha)<1$ を満たす任意の連続的微分可能な関数 $\hat{g}:\mathbb{R}\to\mathbb{R}$ を用いて，

$$\gamma_\mu(\lambda):=\mu\hat{g}(\lambda/\mu) \tag{11}$$

で定義される関数である．ReSNA では (i)–(iii) を満たす関数として，

$$\hat{g}(\alpha):=\frac{1}{2}\bigl(\sqrt{\alpha^2+4}+\alpha\bigr) \tag{12}$$

を採用している．このとき，任意の $\lambda\in\mathbb{R}$ に対して，$\lim_{\mu\searrow 0}\gamma_\mu(\lambda)=\max\{0,\lambda\}$ であることに注意すると，関数 Φ_μ は以下の 2 つの性質を満たすことがわかる：

- 任意の固定された $\mu>0$ に対して，Φ_μ は連続的微分可能；
- 任意の固定された $(x,y)\in\mathbb{R}^n\times\mathbb{R}^n$ に対して $\lim_{\mu\searrow 0}\Phi_\mu(x,y)=\Phi_{\mathrm{NR}}(x,y)$.

したがって，残差関数 Φ_{NR} の代わりに，平滑化関数 Φ_μ を $\mu\searrow 0$ としながらニュートン法などを適用していくことが考えられる．これが平滑化法の基本的なアイディアである．

4.3　正則化法

関数 F，G を直接用いる代わりに，正則化パラメータ $\varepsilon>0$ によって

$$\begin{aligned}F_\varepsilon(x,p)&:=F(x,p)+\varepsilon x,\\ G_\varepsilon(x,p)&:=G(x,p)+\varepsilon p,\end{aligned}$$

で定義された関数 $F_\varepsilon:\mathbb{R}^n\times\mathbb{R}^l\to\mathbb{R}^n$，$G_\varepsilon:\mathbb{R}^n\times\mathbb{R}^l\to\mathbb{R}^l$ を用いるのが正則化法である．一般に関数 F_ε，G_ε は収束性の面において F や G そのものよりも良い性質をもつことが多い．たとえば，関数 $\Gamma=\binom{F}{G}$ が単調[7]であるとき，$\binom{F_\varepsilon}{G_\varepsilon}$ は任意の $\varepsilon>0$ に対して強単調[8]であるため，アルゴリズムがより緩い条件で収束することが期待できる．

4.4　正則化平滑化ニュートン法

(9) で定義された関数 H_{NR} に平滑化と正則化を施した関数 $H_{\mu,\varepsilon}:\mathbb{R}^n\times\mathbb{R}^n\times\mathbb{R}^l\to\mathbb{R}^{2n+l}$ を

$$H_{\mu,\varepsilon}(x,y,p):=\begin{pmatrix}\Phi_\mu(x,y)\\ F_\varepsilon(x,p)-y\\ G_\varepsilon(x,p)\end{pmatrix} \tag{13}$$

で定義する．このとき，$(\mu,\varepsilon)\searrow(0,0)$ としながら，ベクトル方程式 $H_{\mu,\varepsilon}(x,y,p)=0$ をニュートン法で解いていくのが，正則化平滑化ニュートン法である．具体的に平滑化パラメータ μ や正則化パラメータ ε をどのように 0 に収束させていくかについては，次の節で述べる．

5.　アルゴリズム

本節では，ReSNA に用いられているアルゴリズムについて，その具体的な内容と収束性について述べる．より詳しい内容については，[8] も併せて参考にされたい．

5.1　アルゴリズムに用いる付加関数

アルゴリズムにおいて，パラメータの制御を行うために必要な関数を定義する．

(a) 関数 $\tilde{\lambda}:\mathbb{R}^n\to[0,+\infty)$ は以下で定義される：

$$\tilde{\lambda}(z):=\begin{cases}\min\limits_{i\,\in\,\mathcal{I}(z)}|\lambda_i(z)| & (\mathcal{I}(z)\neq\emptyset)\\ 0 & (\mathcal{I}(z)=\emptyset),\end{cases}$$

ここで，$\lambda_i(z)$ $(i=1,2)$ は z のスペクトル値であり，$\mathcal{I}(z)\subseteq\{1,2\}$ は $\mathcal{I}(z):=\{i\,|\,\lambda_i(z)\neq 0\}$ で定義される添字集合である．

(b) 関数 $\overline{\mu}:\mathbb{R}^n\times\mathbb{R}^n\to[0,+\infty]$ は以下で定義される：

$$\overline{\mu}(\alpha,\delta):=\begin{cases}+\infty & (\delta\geq 1/2\ \text{ or }\ \alpha=0)\\ \dfrac{1}{2}|\alpha|\sqrt{\delta} & (\delta<1/2\ \text{ and }\ \alpha\neq 0).\end{cases}$$

(c) 関数 $\gamma_0^+:\mathbb{R}\to\mathbb{R}$ は以下で定義される：

$$\gamma_0^+(\alpha):=\begin{cases}1 & (\alpha>0)\\ \hat{g}'(0) & (\alpha=0)\\ 0 & (\alpha<0).\end{cases}$$

ここで，(11) で定義された関数 γ_μ に対して，任意の α で $\lim_{\mu\searrow 0}\gamma_\mu'(\alpha)=\gamma_0^+(\alpha)$ が成り立つことに注意する．次の命題は，アルゴリズムの収束性を保障するうえで核となるものであり，パラメータ μ を上手く決めることにより $|\gamma_\mu'(\alpha)-\gamma_0^+(\alpha)|$ の上限値が制御できることを示している．

---

[7] 任意の $(z,z')\in\mathbb{R}^{n+l}\times\mathbb{R}^{n+l}$ に対して $\bigl(\Gamma(z)-\Gamma(z')\bigr)^\top(z-z')\geq 0$ が成り立つこと．
[8] ある正の実数 $\varepsilon>0$ が存在して，任意の $(z,z')\in\mathbb{R}^{n+l}\times\mathbb{R}^{n+l}$ に対して $\bigl(\Gamma(z)-\Gamma(z')\bigr)^\top(z-z')\geq\varepsilon\|z-z'\|^2$ が成り立つこと．

命題 5.1. $\alpha \in \mathbb{R}$，$\delta > 0$ を任意の定数とする．また，関数 $\hat{g}$ が (12) で定義されているとする．このとき，任意の $\mu \in (0, \overline{\mu}(\alpha, \delta))$ に対して，以下が成り立つ：

$$|\gamma'_\mu(\alpha) - \gamma_0^+(\alpha)| < \delta. \tag{14}$$

5.2 アルゴリズムとその収束性

本節ではアルゴリズムを具体的に記述し，その収束性について述べる．なお，表記の簡単のため，

$$w^{(k)} := \begin{pmatrix} x^{(k)} \\ y^{(k)} \\ p^{(k)} \end{pmatrix}$$

とする．

アルゴリズム 1.

Step 0 パラメータ $\eta, \rho \in (0,1)$，$\overline{\eta} \in (0, \eta]$，$\sigma \in (0, 1/2)$，$\kappa > 0$，$\hat{\kappa} > 0$，および初期点 $w^{(0)} \in \mathbb{R}^{2n+l}$，$\beta_0 \in (0, \infty)$ を適当に選ぶ．また，$\mu_0 := \|H_{\mathrm{NR}}(w^{(0)})\|$，$\varepsilon_0 := \|H_{\mathrm{NR}}(w^{(0)})\|$，$k := 0$ とする．

Step 1 もし $\|H_{\mathrm{NR}}(w^{(k)})\| = 0$ であれば反復終了．さもなくば，Step 2 へ．

Step 2

Step 2.0 $v^{(0)} := w^{(k)} \in \mathbb{R}^{2n+l}$，$j := 0$ とおく．

Step 2.1 以下のニュートン方程式を解き，その解を $\hat{d}^{(j)} \in \mathbb{R}^{2n+l}$ とする：

$$H_{\mu_k, \varepsilon_k}(v^{(j)}) + \nabla H_{\mu_k, \varepsilon_k}(v^{(j)})^T d = 0.$$

Step 2.2 もし $\|H_{\mu_k, \varepsilon_k}(v^{(j)} + \hat{d}^{(j)})\| \leq \beta_k$ が成り立つならば，$w^{(k+1)} := v^{(j)} + \hat{d}^{(j)}$ として Step 3 へ．さもなくば，Step 2.3 へ．

Step 2.3 次の不等式を満たす最小の非負整数 m を見つける：

$$\begin{aligned} &\left\| H_{\mu_k, \varepsilon_k}(v^{(j)} + \rho^m \hat{d}^{(j)}) \right\|^2 \\ &\qquad \leq (1 - 2\sigma\rho^m) \left\| H_{\mu_k, \varepsilon_k}(v^{(j)}) \right\|^2. \end{aligned}$$

さらに，$m_j := m$，$\tau_j := \rho^{m_j}$，$v^{(j+1)} := v^{(j)} + \tau_j \hat{d}^{(j)}$ とする．

Step 2.4 もし

$$\|H_{\mu_k, \varepsilon_k}(v^{(j+1)})\| \leq \beta_k \tag{15}$$

ならば，$w^{(k+1)} := v^{(j+1)}$ とし Step 3 へ．さもなくば，$j := j + 1$ とし Step 2.1 に戻る．

Step 3 パラメータを次のように更新する：

$$\begin{aligned} \mu_{k+1} &:= \min \Big\{ \kappa \|H_{\mathrm{NR}}(w^{(k+1)})\|^2,\ \mu_0 \overline{\eta}^{k+1}, \\ &\qquad \overline{\mu}\big(\tilde{\lambda}(x^{(k+1)} - y^{(k+1)}), \hat{\kappa}\|H_{\mathrm{NR}}(w^{(k+1)})\|\big) \Big\}, \\ \varepsilon_{k+1} &:= \min \left\{ \kappa \|H_{\mathrm{NR}}(w^{(k+1)})\|^2, \varepsilon_0 \overline{\eta}^{k+1} \right\}, \\ \beta_{k+1} &:= \beta_0 \eta^{k+1}. \end{aligned}$$

$k := k + 1$ とし，Step 1 に戻る．

内部反復 Step 2.0 – 2.4 では，ニュートン法とアルミホのステップサイズルールを組み合わせることにより $\|H_{\mu_k, \varepsilon_k}(w^{(k+1)})\| \leq \beta_k$ を満たす $w^{(k+1)}$ を見つけるという構造になっている．なお，すべての k に対して，有限の j で (15) を満たす $v^{(j+1)} = w^{(k+1)}$ を見つけられることが理論的に証明されている．Step 3 において $\overline{\mu}$, $\tilde{\lambda}$ は前節で定義された関数である．なお，Step 3 ではパラメータの更新ルールを定めているが，$0 < \overline{\eta} \leq \eta < 1$ であることより，$\{\beta_k\}$，$\{\mu_k\}$，$\{\varepsilon_k\}$ のいずれも 0 に収束することに注意する．特に，それらの更新式において $\mu_0 \overline{\eta}^{k+1}$，$\varepsilon_0 \overline{\eta}^{k+1}$，$\beta_0 \eta^{k+1}$ の項はアルゴリズムの大域的収束性を，$\kappa \|H_{\mathrm{NR}}(w^{(k+1)})\|^2$，$\overline{\mu}\big(\tilde{\lambda}(x^{(k+1)} - y^{(k+1)}), \hat{\kappa}\|H_{\mathrm{NR}}(w^{(k+1)})\|\big)$，$\kappa \|H_{\mathrm{NR}}(w^{(k+1)})\|^2$ の項は局所的な 2 次収束性を保障するのに重要な役割を果たしている．

実際，論文 [8] では，SOCCP (1) において $l = 0$，すなわち p と G が存在しない場合の収束解析がなされている．その結論のみを以下に示す．

定理 5.1. SOCCP (1) において $l = 0$，$p \equiv \emptyset$，$G \equiv \emptyset$ とし，関数 F が単調であり，解集合が空でないとする．また，$\{\nabla H_{\mu_k, \varepsilon_k}(w^{(k)})\}$ の任意の集積点が正則であるとする．このとき，アルゴリズム 1 で生成される点列 $\{w^k\}$ は SOCCP (1) の解に大域的かつ 2 次のオーダーで収束する．

謝辞 本稿を執筆する機会をくださりました電気通信大学の村松正和先生をはじめ，編集委員の先生方，OR 学会機関誌編集事務局の方々に御礼申し上げます．

参考文献

[1] http://www.plan.civil.tohoku.ac.jp/opt/hayashi/

[2] Hayashi, S., Yamashita, N. and Fukushima, M., "Robust Nash equilibria and second-order cone complementarity problems," *Journal of Nonlinear and Convex Analysis*, **6**, 283–296, 2005.

[3] Nishimura, R., Hayashi, S. and Fukushima, M., "Robust Nash equilibria in *N*-person noncooperative games: Uniqueness and Reformulation," *Pacific Journal of Optimization*, **5**, 237–259, 2009.

[4] Ito, Y., "Robust Wardrop equilibria in the traffic assignment problem with uncertain data," Master's thesis, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, 2011.

[5] Ordóñez, F. and Stier-Moses, N. E., "Robust Wardrop equilibrium," *Network Control and Optimization, Lecture Notes in Computer Science*, **4465**, 247–256, 2007.

[6] 林俊介, "錐相補性問題の理論と応用 (特集 新世代が切り拓く連続最適化)," オペレーションズ・リサーチ：経営の科学, **59**, 152–158, 2014.

[7] http://ja.wikipedia.org/wiki/

[8] Hayashi, S., Yamashita, N. and Fukushima, M., "A combined smoothing and regularization method for monotone second-order cone complementarity problems," *SIAM Journal on Optimization*, **15**, 593–615, 2005.

[9] Kanno, Y., Martins, J. A. C. and Pinto da Costa, A., "Three-dimensional quasi-static frictional contact by using second-order cone linear complementarity problem," *International Journal for Numerical Methods in Engineering*, **65**, 62–83, 2005.

[10] Sturm, J. F., "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, **11/12**, 625–653, 1999.

[11] Toh, K. C., Tütüncü, R. H. and Todd, M. J., "SDPT3 — a matlab software package for semidefinite programming," *Optimization Methods and Software*, **11**, 545–581, 1999.

[12] Yamashita, M., Fujisawa, K., Fukuda, M., Kobayashi, K., Nakata, K. and Nakata, M., "Latest developments in the SDPA family for solving large-scale SDPs, *Handbook on Semidefinite, Cone and Polynomial Optimization: Theory, Algorithms, Software and Applications*, Anjos, M. F. and Lasserre, J. B. (eds.), Springer, pp. 687–714, 2011.

[13] Faraut, J. and Korányi, A., *Analysis on Symmetric Cones*, Clarendon Press, 1994.

[14] Fukushima, M., Luo, Z.-Q. and Tseng, P., "Smoothing functions for second-order cone complementarity problems," *SIAM Journal on Optimization*, **12**, 436–460, 2001.

[15] 福島雅夫,『非線形最適化の基礎』, 朝倉書店, 2001.

