Juliaによる
数理最適化

博士（理学） 小林　和博【著】

コロナ社

まえがき

本書は，プログラミング言語 Julia を用いて数理最適化問題を解く方法を扱うものである。Julia は科学技術計算で求められる高い計算パフォーマンスと，手軽なプロトタイプ作成環境の両方を実現する言語である。

「数理最適化問題を解く」といっても，本書は問題を解くアルゴリズム，例えば線形最適化問題を解く単体法自体をプログラムとして実現する方法を扱っているのではない。このようなアルゴリズムは既存のパッケージに全面的に頼っている。本書で扱っているのは，解きたい応用上の問題の答えを，これらの既存のパッケージを用いて得る方法である。数理最適化自体の研究ではなく，数理最適化の研究成果を利用して実際の目の前の問題を解くことに興味がある読者を想定している。例えば，新しく読んだ論文に書かれた数理最適化モデルの数値実験を，ささっと目の前のコンピューターで再現できるようになることを想定している。

用いるのは，四則演算と大小比較がほとんどである。行列の記号が出てくるが，本文を読めばわかる通り，実際に行っている演算は積と和である。読者の好みによって線形代数の教科書があれば参考になるかもしれないが，それすら必要ないと思われる。企業で数理最適化を実践されている方々，異動でこれから数理最適化に取り組む必要のある方々，オペレーションズ・リサーチを専門とする学部生・大学院生の方々には便利に使っていただけるはずである。また，意欲的な小中高生にもぜひ読んでいただきたい。ここに書かれていることが自分で実行できるようになれば，数理最適化とオペレーションズ・リサーチの最新の研究の入り口に立ったといえる。その後は，ぜひこの分野の研究者とコンタクトをとり，最新の研究にチャレンジしてみてほしい。

あまり知られてはいないが，数理最適化は，これまでも，そしてこれからも，理工学の基盤を支える大変重要な分野である。最近では機械学習や量子情報理

論の基盤技術となっているが，今後も将来にわたってまだ見ないさまざまな分野の基盤となることは確実である。

いまのわれわれが（超）大規模な数理最適化問題を解くことができるのは，高性能なアルゴリズムと，それを実現するソフトウェアが存在するおかげである。数理最適化のユーザーであるわれわれは，このことを決して忘れてはならない。数万の変数，数万の不等式を難なく扱える線形最適化パッケージがなければ，われわれはそもそも実務規模の線形最適化問題は解けない。このような高い性能は，間違いなく数学的な基礎研究・理論研究に基づいている。その意味で筆者は，数理最適化理論の研究者と，その成果をソフトウェアとして実装し，公開している研究者の方々に最大限の敬意を抱いている。これらの方々が，鋭敏な洞察と緻密さで理論を整備し，ソフトウェアの開発・検証をしてくださるおかげで，われわれユーザーは実務規模の数理最適化問題を解くことができる。

プログラミング言語やそのパッケージの使い方を身につけるには，公式ドキュメントにあたることが最も近道であると考えられる。それは，言語やパッケージの開発者は，それらの特徴を最大限に理解してもらえるようにドキュメントを作成しているからである。したがって，本書の執筆においても Julia とその数理最適化パッケージ JuMP の公式ドキュメントを頻繁に参照し，それらに示されている例を各所で用いた。公式ドキュメントを読み解く以外の王道的な方法は，新しい言語では特に見つかりにくいが，数理最適化に関しては本書がその 1 つになることを期待している。

2023 年 2 月

小林　和博

目　　次

1章　Juliaの利用環境の構築

2章　Juliaの基礎知識

3章 数理最適化の概要

4章 線形最適化問題

5章　グラフ最適化問題

6章　整数最適化問題

7章　2次最適化問題

8章　2次錐最適化問題

9章 半正定値最適化問題

8.3 信号推定問題のよい定式化

7章では，l_0 ノルム制約を l_1 ノルムで緩和することで，信号推定問題を2次最適化問題として定式化した。ここでは，l_0 ノルムを整数変数を用いて定式化することで，よりよい推定信号を得る方法を述べる。

まず，0–1 変数 z_i を新たに導入する。そして，x_i のとりうる最大値 u を用いたつぎの制約式 (8.13) を課す。

$$x_i \leqq u z_i,\ z_i \in \{0,1\} \qquad (i = 1, 2, \cdots, n) \tag{8.13}$$

この式は，z_i が 0 であれば $x_i \leqq 0$ となるので，$x_i = 0$ であることを課していることになる。一方，z_i が 1 であれば $x_i \leqq u$ であるので，x_i は 0 より大きい値をとりうる。したがって，$\boldsymbol{x}$ の非ゼロ要素の数は $\displaystyle\sum_{i=1}^{n} z_i$ で表される。このことから，l_0 ノルムによる疎性制約式 (8.14)

$$\|\boldsymbol{x}\|_0 \leqq k \tag{8.14}$$

は，不等式の組 (8.15) で表されることがわかる。

$$\begin{cases} \displaystyle\sum_{i=1}^{n} z_i \leqq k \\ x_i \leqq u z_i \qquad (i = 1, 2, \cdots, n) \end{cases} \tag{8.15}$$

これより，l_0 ノルムによる疎性制約をもつ信号推定問題は，問題 (8.16) として表される。変数は $\boldsymbol{x}$ と $\boldsymbol{z}$ であり，$\boldsymbol{y}$ は入力データであることに注意する。

$$\begin{cases} \text{最小化} \quad : \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \\ \text{制約条件} : \displaystyle\sum_{i=1}^{n} z_i \leqq k \qquad (k \text{ は正の整数}) \\ \qquad\qquad x_i \leqq u z_i \qquad (i = 1, 2, \cdots, n) \\ \qquad\qquad z_i \in \{0,1\} \qquad (i = 1, 2, \cdots, n) \end{cases} \tag{8.16}$$

この問題は，0-1 変数をもつ 2 次錐最適化問題として定式化することができる。まず，目的関数を展開して式 (8.17) と表す。

$$\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \;=\; \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} 2x_i y_i + \sum_{i=1}^{n} y_i^2 \tag{8.17}$$

ここで，変数 s_i（$i=1,2,\cdots,n$）を新たに導入し，式 (8.17) の最初の和における x_i^2 を s_i で置き換え，制約式に $s_i \geqq x_i^2$ という制約を加える。このようにして，問題 (8.18) を得る。

$$\begin{cases} \text{最小化} \quad : \displaystyle\sum_{i=1}^{n} s_i - \sum_{i=1}^{n} 2x_i y_i + \sum_{i=1}^{n} y_i^2 & \\ \text{制約条件} : \displaystyle\sum_{i=1}^{n} z_i \leqq k & (k \text{ は正の整数}) \\ x_i \leqq u z_i & (i=1,2,\cdots,n) \\ s_i \geqq x_i^2 & (i=1,2,\cdots,n) \\ z_i \in \{0,1\} & (i=1,2,\cdots,n) \end{cases} \tag{8.18}$$

制約式 $s_i \geqq x_i^2$ は 2 次錐制約として表されるので，問題 (8.18) は，0–1 変数をもつ 2 次錐最適化問題 (8.19) として表される。

$$\begin{cases} \text{最小化} \quad : \displaystyle\sum_{i=1}^{n} s_i - \sum_{i=1}^{n} 2x_i y_i + \sum_{i=1}^{n} y_i^2 & \\ \text{制約条件} : \displaystyle\sum_{i=1}^{n} z_i \leqq k & (k \text{ は正の整数}) \\ x_i \leqq u z_i & (i=1,2,\cdots,n) \\ \begin{bmatrix} s_i+1 \\ s_i-1 \\ 2x_i \end{bmatrix} \in \mathcal{K}_3 & (i=1,2,\cdots,n) \\ z_i \in \{0,1\} & (i=1,2,\cdots,n) \end{cases} \tag{8.19}$$

問題 (8.19) を解くためのプログラムが，**プログラム 8-4** である。

プログラム 8-4（信号推定問題を整数 2 次錐最適化問題で解くプログラム）

```
using CSV, DataFrames
using JuMP, Pajarito, HiGHS, SCS
using LinearAlgebra, SparseArrays

noiseless = DataFrame(CSV.File("synthetic_noiseless.csv"));
sigma05 = DataFrame(CSV.File("synthetic_sigma05.csv"));
n, k = 300, 97;
y = Matrix(sigma05[1:n, :]);
yast = Matrix(noiseless[1:n, :]);

oa_solver = optimizer_with_attributes(HiGHS.Optimizer,
    MOI.Silent() => true,
    "mip_feasibility_tolerance" => 1e-8,
    "mip_rel_gap" => 1e-6
);
conic_solver = optimizer_with_attributes(SCS.Optimizer,
    MOI.Silent() => true,
);
opt = optimizer_with_attributes(Pajarito.Optimizer,
    "time_limit" => 600,
    "oa_solver" => oa_solver,
    "conic_solver" => conic_solver,
);

s_socp = Model(opt);
@variable(s_socp, x[1:n] >= 0);
@variable(s_socp, s[1:n] >= 0);
@variable(s_socp, z[1:n], Bin);
@objective(s_socp, Min,
    sum(s[i] - 2y[i]*x[i] + y[i]^2 for i in 1:n)
);
@constraint(s_socp, sum(z[:]) <= k);
@constraint(s_socp,[i in 1:n],
    [s[i]+1; s[i]-1; 2x[i]] in SecondOrderCone()
    );
u=findmax(Matrix(y))[1]*1.1;
@constraint(s_socp, [i in 1:n], x[i] <= u*z[i]);
optimize!(s_socp);
@show termination_status(s_socp);
@show objective_value(s_socp);

tol = 1e-5;
nzcnt = length([i for i in 1:n if value.(x[i]) > tol]);
println("推定信号での非ゼロ要素の数（l0 ノルム）: ", nzcnt);
```

```
45  println("推定信号での絶対値の和（l1 ノルム）: ", norm(value.(x), 1));
46  println("測定データとの誤差: ", norm(value.(x) - y, 2)^2);
```

この問題は整数 2 次錐最適化問題であるので，整数錐最適化ソルバーである Pajarito を用いる。Pajarito をインストールするには，対話型実行環境で**実行例 8.5** に示したコマンドを実行する。

実行例 8.5

```
julia> using Pkg
julia> Pkg.add("Pajarito")
julia> using Pajarito
```

Pajarito は，整数錐最適化問題のソルバーである。このソルバーは，2 つの外部ソルバーを利用する。1 つは整数最適化ソルバーであり，もう 1 つは錐最適化ソルバーである。Pajarito をソルバーとして指定するために，25 行目で Model(opt) と指定している。そして，opt は 19〜23 行目で生成している。opt は，optimizer_with_attributes() で生成するが，その第 1 引数に Pajarito.Optimizer を指定することで，ソルバーとして Pajarito を用いることを定めている。このとき，同時に "oa_solver" と "conic_solver" も指定する。"oa_solver" によって用いる整数最適化ソルバーを，"conic_solver" によって用いる錐最適化ソルバーを，それぞれ指定する。oa_solver は 11〜15 行目で生成している。第 1 引数を HiGHS.Optimizer としているが，これは整数最適化ソルバーとして HiGHS を用いることを定めている。conic_solver は 16〜18 行目で生成している。第 1 引数を SCS.Optimizer としているが，これは錐最適化ソルバーとして SCS を用いることを定めている。

26〜28 行目で変数を定め，29〜31 行目で目的関数を設定している。

32 行目では非ゼロ要素の数を k 以下にするという制約を，33〜35 行目では 2 次錐制約を，それぞれ課している。

36 行目では，x[i] のとりうる値の最大値 u を設定している。ここでは y の最大値の 1.1 倍としている。

37 行目は，x[i] と u[i] の関係を定めるものである。

プログラム 8-4 を実行することで，**実行例 8.6** を得る。

実行例 8.6

```
termination_status(s_socp) = MathOptInterface.OPTIMAL
objective_value(s_socp) = 64.76059625088885
推定信号での非ゼロ要素の数（l0 ノルム）: 97
推定信号での絶対値の和（l1 ノルム）: 170.44195953940974
測定データとの誤差: 64.76175772221698
```

$n = 300$，$k = 97$ とした場合に得られる推定信号（Isocp）を，オリジナルの信号（Noiseless）とあわせて**図 8.2** に示した。オリジナルの信号が 0 であるところは，推定信号でも多くのところで 0 となっている。また，オリジナルの信号が 2 以上の大きな値をとるところでは，推定信号として同程度の値が得られている。

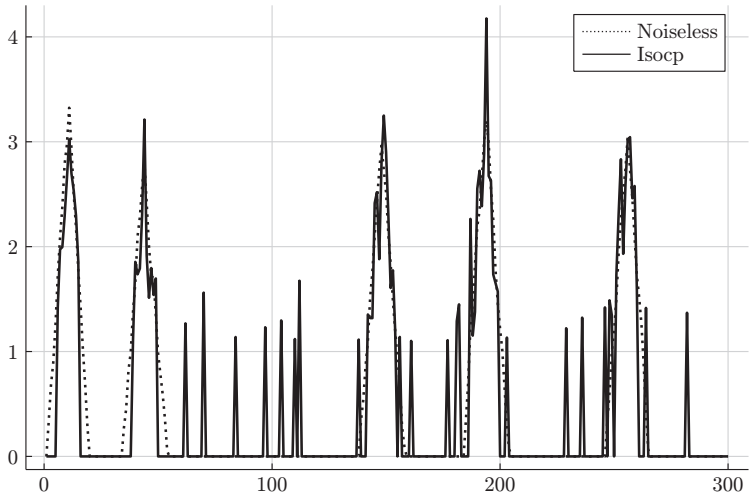

図 8.2　整数 2 次錐最適化問題による推定信号（$n = 300$，$k = 97$）

これらのことから，l_1 ノルムで緩和した定式化で得られる信号（図 7.3，図 7.4）よりもすぐれた推定ができたといえるだろう。l_0 ノルムを緩和せず，そのまま整数最適化問題として定式化すると，このようなすぐれた推定を行うことができる。

索　　引

―― 著 者 略 歴 ――

1998年　東京大学工学部計数工学科卒業
2000年　東京大学大学院工学系研究科修士課程修了（計数工学専攻）
2000年　日本アイ・ビー・エム株式会社勤務
2006年　東京工業大学大学院情報理工学研究科博士課程単位取得退学（数理・計算科学専攻）
2006年　海上技術安全研究所勤務
2009年　博士（理学）（東京工業大学）
2016年　東京理科大学講師
2019年　青山学院大学准教授
　　現在に至る

Julia による数理最適化

Mathematical Optimization with Julia

2023 年 5 月 1 日　初版第 1 刷発行　★

検印省略

著　者　小林和博（こばやし かずひろ）
発行者　株式会社　コロナ社
　　代表者　牛来真也
印刷所　三美印刷株式会社
製本所　有限会社　愛千製本所

112–0011　東京都文京区千石 4–46–10
発行所　株式会社　コロナ社
CORONA PUBLISHING CO., LTD.
Tokyo Japan
振替 00140–8–14844・電話(03)3941–3131(代)
ホームページ　https://www.coronasha.co.jp

ISBN 978–4–339–02934–5　C3055　Printed in Japan　（谷口）