


# 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してほしい。

あまり知られてはいないが、数理最適化は、これまで、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

## ii ま え が き

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の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ですが、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

---

1.1 Julia のインストール	1
1.2 Julia でのパッケージ管理	9
1.3 モデラーとソルバー	10

## 2 章 Juliaの基礎知識

---

2.1 ベクトル	14
2.2 行列	17
2.2.1 密行列の表現	17
2.2.2 疎行列の表現	19
2.3 線形演算	21
2.3.1 ベクトルとスカラーの演算	21
2.3.2 ベクトル同士の演算	22
2.3.3 行列とベクトルの積	23
2.3.4 行列同士の積	24
2.4 反復処理	25

## 3章 数理最適化の概要

---

3.1 変数と定数	29
3.2 目的関数	29
3.3 制約条件	30
3.4 数理最適化問題の分類	30
3.4.1 凸最適化問題と非凸最適化問題	30
3.4.2 線形最適化問題	31
3.4.3 2次最適化問題	32
3.4.4 整数最適化問題	33
3.4.5 半正定値最適化問題	34

## 4章 線形最適化問題

---

4.1 問題例：栄養素の問題	35
4.2 変数と定数	37
4.3 目的関数	37
4.4 制約条件	38
4.5 定式化	40
4.6 解法	40
4.6.1 単体法	41
4.6.2 内点法	41
4.6.3 プログラムを用いた求解	42
4.6.4 実行可能な問題例の生成	51
4.7 双対問題	54
4.8 潜在価格	59
4.9 輸送最適化問題	61

## 5章 グラフ最適化問題

---

5.1 グラフの表現	77
5.2 最短路問題	79
5.2.1 整数最適化問題として解く方法	80
5.2.2 ダイクストラ法を用いて解く方法	92
5.3 最大流問題	95

## 6章 整数最適化問題

---

6.1 集合分割問題	103
6.2 巡回セールスマン問題	117
6.2.1 不完全な定式化	119
6.2.2 完全な定式化	121
6.3 施設配置問題	125

## 7章 2次最適化問題

---

7.1 ソルバーを用いて解く方法	136
7.2 信号推定問題	137

## 8章 2次錐最適化問題

---

8.1 2次錐と2次錐最適化問題	145
8.2 多項式最適化問題	150
8.3 信号推定問題のよい定式化	156

## 9章 半正定値最適化問題

---

9.1 最大カット問題の緩和解 .....	169
9.2 データ分類問題 .....	185
9.2.1 線形分類 .....	185
9.2.2 2次関数による分類 .....	189
引用・参考文献 .....	194
索引 .....	196

### 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。

$$x_i \leq uz_i, \quad z_i \in \{0, 1\} \quad (i = 1, 2, \dots, n) \quad (8.13)$$

この式は、 $z_i$  が0であれば  $x_i \leq 0$  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  $x_i \leq u$  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、 $\mathbf{x}$  の非ゼロ要素の数は  $\sum_{i=1}^n z_i$  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)

$$\|\mathbf{x}\|_0 \leq k \quad (8.14)$$

は、不等式の組 (8.15) で表されることがわかる。

$$\begin{cases} \sum_{i=1}^n z_i \leq k \\ x_i \leq uz_i \quad (i = 1, 2, \dots, n) \end{cases} \quad (8.15)$$

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  $\mathbf{x}$  と  $\mathbf{z}$  であり、 $\mathbf{y}$  は入力データであることに注意する。

$$\begin{cases} \text{最小化} & : \|\mathbf{x} - \mathbf{y}\|_2^2 \\ \text{制約条件} & : \sum_{i=1}^n z_i \leq k \quad (k \text{ は正の整数}) \\ & x_i \leq uz_i \quad (i = 1, 2, \dots, n) \\ & z_i \in \{0, 1\} \quad (i = 1, 2, \dots, n) \end{cases} \quad (8.16)$$



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。

$$\|\mathbf{x} - \mathbf{y}\|_2^2 = \sum_{i=1}^n x_i^2 - \sum_{i=1}^n 2x_i y_i + \sum_{i=1}^n y_i^2 \quad (8.17)$$

ここで、変数  $s_i$  ( $i = 1, 2, \dots, n$ ) 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  $s_i \geq x_i^2$  という制約を加える。このようにして、問題 (8.18) を得る。

$$\left\{ \begin{array}{l} \text{最小化} : \sum_{i=1}^n s_i - \sum_{i=1}^n 2x_i y_i + \sum_{i=1}^n y_i^2 \\ \text{制約条件} : \sum_{i=1}^n z_i \leq k \quad (k \text{ は正の整数}) \\ \quad \quad \quad x_i \leq uz_i \quad (i = 1, 2, \dots, n) \\ \quad \quad \quad s_i \geq x_i^2 \quad (i = 1, 2, \dots, n) \\ \quad \quad \quad z_i \in \{0, 1\} \quad (i = 1, 2, \dots, n) \end{array} \right. \quad (8.18)$$

制約式  $s_i \geq x_i^2$  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。

$$\left\{ \begin{array}{l} \text{最小化} : \sum_{i=1}^n s_i - \sum_{i=1}^n 2x_i y_i + \sum_{i=1}^n y_i^2 \\ \text{制約条件} : \sum_{i=1}^n z_i \leq k \quad (k \text{ は正の整数}) \\ \quad \quad \quad x_i \leq uz_i \quad (i = 1, 2, \dots, n) \\ \quad \quad \quad \begin{bmatrix} s_i + 1 \\ s_i - 1 \\ 2x_i \end{bmatrix} \in \mathcal{K}_3 \quad (i = 1, 2, \dots, n) \\ \quad \quad \quad z_i \in \{0, 1\} \quad (i = 1, 2, \dots, n) \end{array} \right. \quad (8.19)$$

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)

```

1  using CSV, DataFrames
2  using JuMP, Pajarito, HiGHS, SCS
3  using LinearAlgebra, SparseArrays
4
5  noiseless = DataFrame(CSV.File("synthetic_noiseless.csv"));
6  sigma05 = DataFrame(CSV.File("synthetic_sigma05.csv"));
7  n, k = 300, 97;
8  y = Matrix(sigma05[1:n, :]);
9  yast = Matrix(noiseless[1:n, :]);
10
11  oa_solver = optimizer_with_attributes(HiGHS.Optimizer,
12      MOI.Silent() => true,
13      "mip_feasibility_tolerance" => 1e-8,
14      "mip_rel_gap" => 1e-6
15  );
16  conic_solver = optimizer_with_attributes(SCS.Optimizer,
17      MOI.Silent() => true,
18  );
19  opt = optimizer_with_attributes(Pajarito.Optimizer,
20      "time_limit" => 600,
21      "oa_solver" => oa_solver,
22      "conic_solver" => conic_solver,
23  );
24
25  s_socp = Model(opt);
26  @variable(s_socp, x[1:n] >= 0);
27  @variable(s_socp, s[1:n] >= 0);
28  @variable(s_socp, z[1:n], Bin);
29  @objective(s_socp, Min,
30      sum(s[i] - 2y[i]*x[i] + y[i]^2 for i in 1:n)
31  );
32  @constraint(s_socp, sum(z[:]) <= k);
33  @constraint(s_socp, [i in 1:n],
34      [s[i]+1; s[i]-1; 2x[i]] in SecondOrderCone()
35  );
36  u=findmax(Matrix(y))[1]*1.1;
37  @constraint(s_socp, [i in 1:n], x[i] <= u*z[i]);
38  optimize!(s_socp);
39  @show termination_status(s_socp);
40  @show objective_value(s_socp);
41
42  tol = 1e-5;
43  nzcnt = length([i for i in 1:n if value.(x[i]) > tol]);
44  println("推定信号での非ゼロ要素の数 (10 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。

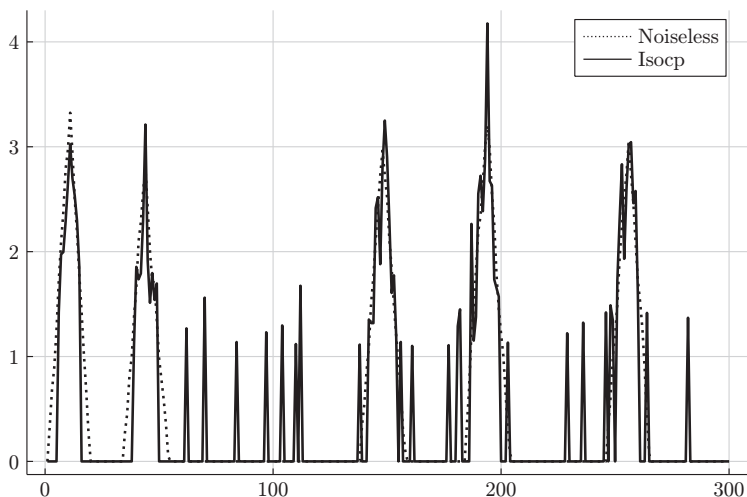


図 8.2 整数 2 次錐最適化問題による推定信号 ( $n = 300$ ,  $k = 97$ )

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

<b>【あ】</b>		最適解	28			<b>【た】</b>	
値	66	最適値	28			ダイクストラ法	92
アフィン関数	32	<b>【し】</b>				対話型	3
アフィン表現	48	辞書	66			ターミナル	3
<b>【え】</b>		施設配置問題	125			単体法	40
枝	77	実行可能解	28			端点	77
<b>【か】</b>		実行可能多面体	41			<b>【て】</b>	
カット	170	実行可能領域	28			定数	29
環境変数	3	集合分割問題	103			データフレーム	71
完全グラフ	118	主問題	56			点	77
<b>【き】</b>		巡回セールスマン問題	117			<b>【と】</b>	
キー	66	巡回路	118			等式制約	30
擬似乱数の種	62	<b>【す】</b>				特異値分解	178
行列	17	錐	146			凸	31
行列変数	164	錐最適化問題	147			凸関数	31
<b>【く】</b>		スライシング	15			凸最適化問題	31
グラフ	77	<b>【せ】</b>				凸集合	31
<b>【け】</b>		整数最適化問題	33, 102			凸錐	146
決定変数	28, 36	整数条件	33, 102			トレース	164
<b>【こ】</b>		整数変数	33			<b>【な】</b>	
固定費	126	正方行列	165			内積	23, 38, 164
<b>【さ】</b>		積	21, 22, 23, 24			内点法	40
最大カット問題	169	線形最適化問題	31, 36			内包表記	26
最大流	95	潜在価格	59			流れ	95
最大流問題	95	<b>【そ】</b>				<b>【は】</b>	
最短路問題	79	双対問題	56			配送計画問題	107
		添え字	15			パス	79
		疎行列	19			パッケージ管理	9
		ソルバー	11			半正定値	34

半正定値最適化緩和	172
半正定値最適化問題	34, 164
<b>【ひ】</b>	
非凸最適化問題	31
非負象限	145
非負条件	36, 145
<b>【ふ】</b>	
不等式制約	30
部分巡回路	121
負閉路	92
フロー保存則	83
分割	103
分枝限定法	33

<b>【へ】</b>	
ベクトル	14
変数	29
<b>【み】</b>	
密行列	17
<b>【む】</b>	
無向グラフ	77
<b>【め】</b>	
メタヒューリスティクス	117
メルセンヌツイスター	111

<b>【も】</b>	
目的関数	28, 29
モデラー	11
<b>【ゆ】</b>	
有向グラフ	77
輸送最適化問題	61
<b>【よ】</b>	
容量	95
容量制約	96
<b>【ら】</b>	
ラプラシアン	172

<b>【A】</b>	
Array	49
<b>【B】</b>	
big-M	123
Bin	81
<b>【C】</b>	
copy()	17
CSV	71
<b>【D】</b>	
DataFrames	71
DenseAxisArray	50
Dict()	66
DiGraph	100
dijkstra_shortest_path()	92
DijkstraStates	93
<b>【E】</b>	
enumerate()	67

enumerate_paths()	94
<b>【F】</b>	
findnz()	87
Float64	14, 18
for	25
<b>【G】</b>	
Graphs	77, 92
GraphsFlows	100
<b>【H】</b>	
HiGHS	13, 42
<b>【I】</b>	
inneighbors()	91
Int64	15
<b>【K】</b>	
keys()	69
<b>【L】</b>	
LinearAlgebra.diag()	175
LinearAlgebra.dot()	82

LinearAlgebra.norm()	110, 129
<b>【M】</b>	
Matrix	17
Max	43
maximum_flow()	100
MersenneTwister	111
Min	43
Model()	42
MOI.NormOneCone()	139
<b>【O】</b>	
objective.value()	44
OPTIMAL	44
optimize()	44
outneighbors()	91
<b>【P】</b>	
perspective 再定式化	162
Pkg	9
Plots	114
Plots.plot()	114
positive semidefinite	166

			SparseAxisArray	50		
			sum()	120	<b>[Z]</b>	
			svd()	178	zip()	27
					<b>[記号, 数字]</b>	
	<b>[R]</b>		<b>[T]</b>		@constraint()	43
randperm()	111		termination_status()	44	@objective()	43
reduce	89		trunc()	110	@variable()	42
round()	176				2次最適化問題	32
			<b>[V]</b>		2次錐最適化問題	145
	<b>[S]</b>		Vector	14		
scatter()	130					
SCS	13, 136					
SimpleWeightedDiGraph	89					
SparseArrays	20					

—— 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

© Kazuhiro Kobayashi 2023

2023年5月1日 初版第1刷発行



検印省略

著者 小林 和博  
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)



<出版者著作権管理機構 委託出版物>

本書の無断複製は著作権法上での例外を除き禁じられています。複製される場合は、そのつど事前に、出版者著作権管理機構（電話 03-5244-5088, FAX 03-5244-5089, e-mail: info@jcopy.or.jp）の許諾を得てください。

本書のコピー、スキャン、デジタル化等の無断複製・転載は著作権法上での例外を除き禁じられています。購入者以外の第三者による本書の電子データ化及び電子書籍化は、いかなる場合も認めていません。落丁・乱丁はお取替えいたします。