複素積分によるスペクトル分解を応用した最適化問題の解法

M2019SS002 肥田照久
指導教員：小市俊悟

1　はじめに

既存の最適化手法の一つにニュートン法がある．ニュートン法では，目的関数のヘッセ行列を求めて，それを利用して探索方向を計算する [1]．しかし，目的関数が凸でない場合，ヘッセ行列を用いて求めた探索方向に解を移動させても必ずしも関数値が小さくなるとは限らない．ヘッセ行列が正定値とは限らないからである．このような問題に対処するために，ヘッセ行列を，その固有値のうち，正の固有値に対応する射影行列の和で置き換えて探索方向を計算する方法が考案されている．これにより，目的関数が凸とは限らない場合でも適切な探索方向の計算が行える [2]．

本研究では，このようなヘッセ行列を正の固有値に対応する射影行列の和に置き換えるための計算において，複素積分を利用することで，ニュートン法では対応できない関数でも最適化できるようにする．厳密な複素積分では相応の計算量が必要となるが，ヘッセ行列がそもそも近似的であることを踏まえて，複素積分を和で近似し，実際の計算量を一定程度に収める．これにより非凸関数の最小化についても，実用的な計算速度と安定性を備えたアルゴリズムとする．提案するアルゴリズムは，ヘッセ行列に正の固有値が少ないとき，自動で最急降下法に近い振る舞いをするとも考えられ，利用者が特別な命令をする必要がない点も提案するアルゴリズムの特徴となる．文献 [1] に例示されていた関数に対して，提案アルゴリズムを適用したところ，全体の計算時間に関しては記述がないので分からないが，同じ初期解から十分な精度で最適解に達するまでの反復回数だけに着目すると，文献 [1] で示されていた最急降下法や準ニュートン法が必要とした反復回数よりも少ない回数で提案アルゴリズムは最適解に達することが確認できた．

2　複素積分によるスペクトル分解

提案するアルゴリズムでは，複素積分によるヘッセ行列のスペクトル分解を利用する．

対称行列 A の固有値 λ_i に対応する射影行列を P_i とすると，A は以下のように分解される．

$$A = \sum_i \lambda_i P_i \tag{1}$$

これを対称行列 A のスペクトル分解と呼ぶ [3]．

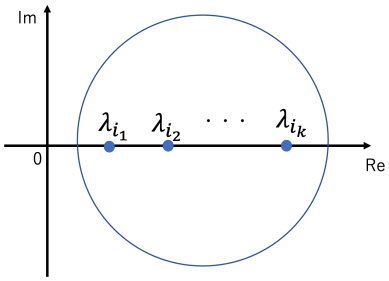

図 1　固有値の分布と積分経路 C

積分経路 C が図 1 のように対称行列 A の固有値 $\lambda_{i_1}, \cdots, \lambda_{i_k}$ を含むとき，下記の複素積分により，積分領域に含まれる固有値に対応する射影行列の和 P が得られる．

$$P = \frac{1}{2\pi i} \oint_C (zI - A)^{-1} dz = \sum_k P_{i_k} \tag{2}$$

このとき，固有値 $\lambda_{i_1}, \cdots, \lambda_{i_k}$ は積分経路 C に含まれていることさえ確かであれば具体的に求めておく必要はない．

さらに，射影行列の性質 $P_i^2 = P_i$，$P_i P_j = O\ (i \neq j)$ から以下が成り立つ．

$$AP = (\sum_i \lambda_i P_i)(\sum_k P_{i_k}) = \sum_k \lambda_{i_k} P_{i_k} \tag{3}$$

したがって，積分経路 C を適切に設定することにより複素積分 (2) を用いて，対称行列 A のスペクトル分解の部分和を得ることができる．

3　提案する最適化アルゴリズム

本研究で提案するアルゴリズムを以下に示す．着想を示すために，下記では，解析的な計算がすべて可能であるとして説明する．

複素積分によるスペクトル分解を用いた最適化アルゴリズム

$\boldsymbol{x} = (x, y, \cdots, z)$ とし，関数 $f(\boldsymbol{x})$ を考える．

0. 初期解 $\boldsymbol{x}$ を与える
1. f の勾配 $\nabla f(\boldsymbol{x})$ とヘッセ行列 $A = \nabla^2 f(\boldsymbol{x})$ を求める
2. A の最大固有値 $\lambda_{\max}$ を求める
3. 次の式で M^+ を計算する

$$M^+ = \frac{1}{2\pi i} \oint_C (zI - A)^{-1} dz \tag{4}$$

C ：$\lambda_{\max}$を中心とする半径 r の円

4. $B = AM^+$ を計算する
5. 適切な $\varepsilon > 0$ を用いて，$B = B + \varepsilon I$（I は単位行列）とする
6. 方程式 $B\boldsymbol{d} = -\nabla f(\boldsymbol{x})$ を解き，$\boldsymbol{d}$ を求める
7. 適切な $\alpha > 0$ を求めて $\boldsymbol{x}$ を $\boldsymbol{x} := \boldsymbol{x} + \alpha\boldsymbol{d}$ に更新し，1 に戻る

1 から 7 までを一通り実行することを 1 反復と数える．アルゴリズムの終了条件は，事前に決めておいた反復回数で終了するように設定している．

上記のアルゴリズムにおいて，行列 B は，ヘッセ行列 A に最も近いような（半）正定値行列であることが望ましいと一般に考えられる．そのような行列 B を得るためには，例えば，A の（正の）最大固有値 $\lambda_{\max}$ を求め，さらに半径 $r = \lambda_{\max}$ として，式 (2) における積分経路 C を設定すれば，4 で得る $B = AM^+$ が求める行列となる．つまり，最大固有値 $\lambda_{\max}$ がわかり，かつ半径 $r = \lambda_{\max}$ の複素積分が可能であれば，A の他の固有値を具体的に知ることなく，最も望ましい行列 B が得られる．これが，上記のアルゴリズムにおいて，最大固有値 $\lambda_{\max}$ を利用する理由である．しかし，実際には，次節で説明するように複素積分を和で近似するため，半径 $r = \lambda_{\max}$ とするのは，必ずしも適切であるとは限らない．半径 r が大きいと，和をとる点が積分経路上に粗く分布することになるので，近似精度が下がると一般には考えられるからである．また，5 で単位行列を加えるのは，4 で得られる B が一般には正定値行列とは限らないためであり，単位行列の ε 倍を加えることで正則行列としている．

4　アルゴリズムの実装方針

MATLAB を利用して 3 節に示したアルゴリズムをプログラムとして実装する．行列やベクトルの基本的な積や逆行列の計算には MATLAB に用意されている関数も使用し，以下のように実装した．

(1) 勾配やヘッセ行列を差分近似で置き換える．
(2) べき乗法により（近似）ヘッセ行列の（最大）固有値を求める．
(3) 実対称行列に対して，複素積分により，その正の固有値に対応する射影行列を計算する．その際，計算量をおさえるために複素積分は，和により近似的に求める．
(4) 探索方向にどれくらい進むべきかを求めるために，すなわち，7 の α を求めるために二分探索法を用いる．

上記の方針に従ってプログラムを作成する．(1)，(2)，(4) については一般的な方法でもあるので，(3) について下記に説明する．

複素積分を次のように和で近似する．

$$M^+ = \frac{1}{2\pi i}\oint_C (zI - A)^{-1}dz \approx \frac{1}{2\pi i}\sum_{j=1}^{N}(z_j I - A)^{-1}(z_{j+1} - z_j) \qquad (5)$$

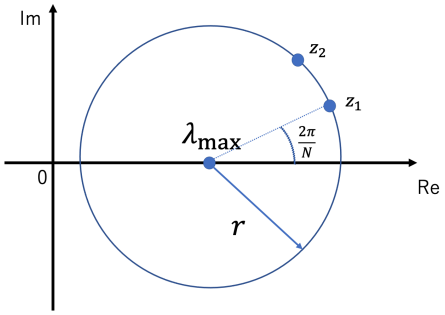

図 2　複素積分を和で近似する際に用いた積分経路上の点

ここで，各 $j = 1, \ldots, N$ について，z_j は図 2 に示すような最大固有値 $\lambda_{\max}$ を用いて $z_j = \lambda_{\max} + r \cdot \exp\left(j\frac{2\pi i}{N}\right)$ と表される点である．z_{N+1} は積分経路上を一周し，z_1 に等しい．本研究では N として，$N = 100$ または $N = 1000$ に設定する．

積分経路となる円の半径 r は，3 節で述べたように，$r = \lambda_{\max}$ とすることが基本であるが，$\lambda_{\max}$ が大きいと，積分の近似精度が良くないことが想定される．そのため，$\lambda_{\max}$ が一定以上の値の場合には $r = 1$ として計算することにした．具体的には N を用いて $\lambda_{\max} > 10N^2$ のときは $r = 1$ とした．同じ実対称行列を対象に最適化アルゴリズムの 5 で定義される行列 B を求めたところ，$N = 100$ の場合と $N = 1000$ の場合で，各成分の差の絶対値は 10^{-3} 程度に収まった．

5　計算量について

提案する最適化アルゴリズムを 4 節の方針に従って実装した場合の計算量について説明する．目的関数の変数の数を n とする．このとき，ヘッセ行列は $n \times n$ の正方行列である．

最大固有値 $\lambda_{\max}$ を求めるのに，べき乗法を用いた場合，1 反復あたり行列とベクトルの積に $\mathrm{O}(n^2)$ の計算量を要する．べき乗法の反復回数は，行列の大きさ程度が目安とも言われるので，ここでは，$\mathrm{O}(n^3)$ 程度であるとする．

複素積分の計算では，逆行列の計算に $\mathrm{O}\left(\frac{n^3}{3}\right)$ の計算量が必要となる．このような逆行列の計算を N 回だけ繰り返すことになるので，最適化アルゴリズムの 3 全体で，$\mathrm{O}\left(\frac{Nn^3}{3}\right)$ の計算量が必要となる．N は，n とは無関係に決めることができるので，N を定数とする場合，$\mathrm{O}\left(\frac{n^3}{3}\right)$

となる．

最適化アルゴリズムの 4 では，行列の積 $B = AM^{+}$ を計算するが，行列の積であるので，素朴に $\mathrm{O}(n^3)$ とする．

最適化アルゴリズムの 6 では，方程式 $B\boldsymbol{d} = -\nabla f(\boldsymbol{x})$ を解く．一次方程式の解法には，様々なものがあるが，計算量はいずれも $\mathrm{O}(n^3)$ 程度となる．

以上より，提案する最適化アルゴリズムの計算量は，1 反復あたり $\mathrm{O}\left(\dfrac{Nn^3}{3}\right)$ もしくは，N を定数として，$\mathrm{O}(n^3)$ となる．通常のニュートン法においても，一次方程式を解く必要があるため，1 反復あたり，$\mathrm{O}(n^3)$ の計算量を要する．したがって，複素積分の近似に伴う N の取り扱いにも依存するが，適用可能な目的関数の範囲も広いことを考慮すれば，計算量が通常のニュートン法に対して極端に大きいわけではないと考える．

6　計算実験

提案する最適化アルゴリズムを用いて計算実験を行った．はじめに関数，初期値，反復回数の上限を設定する．関数は 10 変数で最小値がわかっているものを用いた．実行後，得られた結果と正しい最小値を比較し，その差を調べた．

6.1　積分経路上にとる点の数の違いによる比較

以下の関数と初期値を用いて，複素積分を和で近似する際，積分経路上にとる点の数 N を $N = 100$ と $N = 1000$ に設定した場合について，結果を比較する．

・関数

$$f_1(\boldsymbol{x}) = \exp(x_1^2 + x_2^2) + \exp(x_2^2 + x_3^2) + \cdots \\ \cdots + \exp(x_9^2 + x_{10}^2)$$

$$f_2(\boldsymbol{x}) = x_1(x_1-1)(x_1-2)(x_1-3) + \cdots \\ \cdots + x_{10}(x_{10}-1)(x_{10}-2)(x_{10}-3)$$

$$f_3(\boldsymbol{x}) = x_1^2 + (x_2-1)^2 + (x_3-2)^2 + \cdots + (x_{10}-9)^2$$

$$f_4(\boldsymbol{x}) = \exp(x_1^2) + \exp(x_2-1)^2 + \cdots + \exp(x_{10}-9)^2$$

$$f_5(\boldsymbol{x}) = \exp\left(x_1^2 + (x_2-1)^2 + \cdots + (x_{10}-9)^2\right)$$

・初期値

$$\boldsymbol{x}_1 = [\ 1\quad 2\quad 3\quad 4\quad 5\quad 6\quad 7\quad 8\quad 9\quad 10\ ]$$
$$\boldsymbol{x}_2 = [\ 1\quad 2\quad 3\quad 1\quad 2\quad 3\quad 1\quad 2\quad 3\quad 1\ ]$$
$$\boldsymbol{x}_3 = [\ 10\quad 10\quad 10\quad 10\quad 10\quad 10\quad 10\quad 10\quad 10\quad 10\ ]$$
$$\boldsymbol{x}_4 = [\ 5\quad 10\quad 7\quad 8\quad 6\quad 3\quad 2\quad 9\quad 4\quad 1\ ]$$

表 1 と表 2 は，100 回を反復回数の上限としたとき，正しい最小値に収束したと考えられるのが何回目の反復であったかを示し，100 回で収束しなかった場合は - と表す．表 3 は，各関数が初期値 $\boldsymbol{x}_1$，$N = 1000$ において，収束した際の目的関数値を反復回数とともに示している．

関数や初期値によって収束までの反復回数に差はあるが，ほとんどの場合で $N = 100$ より $N = 1000$ の方が早く収束するか同じ回数で収束した．また，100 回で収束しなかった場合も関数値が小さくなるように解は更新されていた．

表 1　反復 100 回以内に収束した際に，収束までに要した反復回数 ($N = 100$)

	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_4$
$f_1(\boldsymbol{x})$	-	6	14	12
$f_2(\boldsymbol{x})$	8	5	9	9
$f_3(\boldsymbol{x})$	2	2	2	2
$f_4(\boldsymbol{x})$	5	-	-	-
$f_5(\boldsymbol{x})$	15	7	11	8

表 2　反復 100 回以内に収束した際に，収束までに要した反復回数 ($N = 1000$)

	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_4$
$f_1(\boldsymbol{x})$	16	7	17	12
$f_2(\boldsymbol{x})$	9	5	9	9
$f_3(\boldsymbol{x})$	2	2	2	2
$f_4(\boldsymbol{x})$	5	-	-	-
$f_5(\boldsymbol{x})$	15	7	5	8

表 3　初期値 $\boldsymbol{x}_1$，$N = 1000$ で各関数が収束した際の反復回数と目的関数値

関数	反復回数	目的関数値	厳密解
$f_1(\boldsymbol{x})$	16	9.0000	9
$f_2(\boldsymbol{x})$	9	-10.0000	-10
$f_3(\boldsymbol{x})$	2	2.4322×10^{-6}	0
$f_4(\boldsymbol{x})$	5	-10.0000	-10
$f_5(\boldsymbol{x})$	15	1.00000	1

関数 $f_1(\boldsymbol{x})$ で初期値 $\boldsymbol{x}_2$ の場合，関数 $f_1(\boldsymbol{x})$ で初期値 $\boldsymbol{x}_3$ の場合，関数 $f_2(\boldsymbol{x})$ で初期値 $\boldsymbol{x}_1$ の場合に $N = 1000$ より $N = 100$ の方が反復回数が例外的に少ない．しかし，大きな差ではなく，同程度と見なすことができる．関数 $f_1(\boldsymbol{x})$ で初期値 $\boldsymbol{x}_1$ の場合について，$N = 100$ では 100 回の反復では収束しなかったが，$N = 1000$ では 16 回で収束した．関数 $f_5(\boldsymbol{x})$ で初期値 $\boldsymbol{x}_3$ の場合については，どちらも 100 回以内の反復で収束したが，$N = 100$ では 11 回に対し，$N = 1000$ では 5 回と半分以下の反復回数で収束した．

また，表 3 から，提案するアルゴリズムが，計算の精度も十分に最適解を求められていると言える．

6.2　積分経路の違いによる比較

同じ関数と初期値を用いて，積分経路を円ではなく，図 3 のように最大固有値を含む長方形にした計算も行なった．

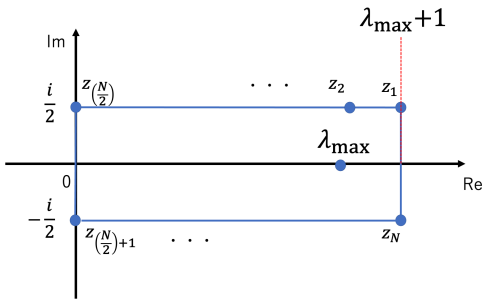

図 3　最大固有値を含む長方形の積分経路

結果として積分経路が円の場合に比べ，100 回以内に収束しない場合が増えた．収束した場合でも，そのほとんどで積分経路を円にして計算したときより，収束までの反復回数が多かった．

6.3　変数の増加による比較

6.1 節に示した関数と同様の関数で変数を 20 個にした関数についても計算実験を行った．

少ない反復回数で収束する場合は $N = 100$ でも $N = 1000$ でもほとんど変わらない反復回数で収束したが，一部の関数や初期値においては，$N = 1000$ の場合でも 100 回の反復では収束しなかった．これは，収束しなかった関数は指数関数であり，初期値において，関数値ならびにヘッセ行列の成分が大きい関数である．このために，逆行列の計算等が行えなくなり，収束しなかったのでないかと考える．このような関数は，そもそも数値計算全般において，取り扱いにくいと考えられる．

6.4　密なヘッセ行列に対する結果

また，以下のような相異なる変数を互いにかけた項を含む関数についても実験を行なった．このような関数はヘッセ行列が密になると考えられる．これらの関数においても正しい挙動かつ，10 回程度の少ない反復回数で収束した．

・関数

$$f_6(\boldsymbol{x}) = x_1(x_1-1)(x_1-2)(x_1-3)\times\cdots \\ \cdots\times x_{10}(x_{10}-1)(x_{10}-2)(x_{10}-3)$$

$$f_7(\boldsymbol{x}) = x_1^2\times(x_2-1)^2\times(x_3-2)^2\times\cdots\times(x_{10}-9)^2$$

6.5　文献にある関数に対する計算実験

次の関数は文献 [1] に掲載されていた関数である．文献 [1] にはこの関数に対して，最急降下法，準ニュートン法を適用した場合に，最適解に収束するまでに必要となった反復回数を示す表があったので，それと本研究で提案するアルゴリズムによる反復回数を比較することにした．全体の計算時間については文献 [1] に示されていなかったので比較できない．

$$f(\boldsymbol{x}) = (x_1-1)^2 + 10\left(x_1^2-x_2\right)^2 \qquad (6)$$

初期値を $\boldsymbol{x} = [\,0\ 1\,]$ として，最適解 $\boldsymbol{x} = [\,1\ 1\,]$，目的関数値 $f(\boldsymbol{x}) = 0$ に収束したときの反復回数と解を表 4 に示す [1].

表 4　式 (6) の関数に対する手法ごとの収束するまでの反復回数と解

手法	反復回数	解
提案手法 ($N = 100$)	8	$\boldsymbol{x} = [\,1.00000\ 1.00000\,]$
提案手法 ($N = 1000$)	4	$\boldsymbol{x} = [\,1.00000\ 1.00000\,]$
最急降下法	500	$\boldsymbol{x} = [\,0.99838\ 0.99669\,]$
準ニュートン法	9	$\boldsymbol{x} = [\,1.00000\ 1.00000\,]$

表 4 から最急降下法は他の手法に比べて非常にゆっくりと収束することがわかる．提案するアルゴリズムにおいては，$N = 100$ ならば 8 回，$N = 1000$ ならば 4 回と準ニュートン法より少ない反復回数で収束した．よって，提案するアルゴリズムを用いた場合，関数や設定する N 次第では，既存の手法と同様かそれ以下の反復回数で収束することもあると言える．

7　おわりに

本研究では，制約なし最適化問題の解法として，ニュートン法におけるヘッセ行列を射影行列の和で置き換え，それを複素積分を利用して計算する最適化アルゴリズムを提案した．提案アルゴリズムを実装し，計算実験を行ったところ，提案した最適化アルゴリズムが，様々な関数に対して少ない反復回数で最小化できることや，目的関数が凸でない場合にも適用できることが確かめられた．

また，同じ関数に対して既存の手法を用いた場合の結果との比較から，設定次第で既存の手法と同等かそれ以下の反復回数で収束すると言え，より効率的に問題を解く有効な手法になり得ることが示せた．また，6 節の実験結果から，ヘッセ行列や複素積分の和の近似による計算精度も十分に保つことができていると考える．

参考文献

[1] 福島雅夫，『新版　数理計画入門』，朝倉書店，2011.

[2] John J. Spitzer,　A numerically stable and efficient technique for the maintenance of positive definiteness in the Hessian for Newton-type methods, *Journal of Computational and Applied Mathematics*, Vol. 3, No. 2, 1977.

[3] 伊理正夫，『一般線形代数』，岩波書店，2003.