日本オペレーションズ・リサーチ学会
2019年 秋季研究発表会

非線形最適化問題に対する任意精度計算可能な内点法の性能評価

05001039　関西大学大学院　*崎山 皓瑛　SAKIYAMA Koei
01308414　関西大学　檀 寛成　DAN Hiroshige

1. はじめに

非線形最適化問題（NLP) には，数値的性質の悪い問題，すなわち，理論的には解けるはずだが，求解中に数値的条件が悪くなることで最適解を求めることができない，あるいは求解までに長い時間が必要となるような NLP が存在する．

我々のグループでは，数値的性質の悪い NLP を解くために，任意精度計算が可能なソフトウェア群 NOA (Nonlinear Optimizer with Arbitrary precision arithmetic) を開発している（例えば [2]）．先行研究 [3] では NLP に対する内点法 “noa_ip” を実装したが，実装に不十分な箇所があった他，任意精度計算を用いた性能評価を行うことができていなかった．そこで本研究では，noa_ip の実装改良を行うとともに，任意精度計算を用いた性能評価，特に倍精度計算との比較を行う．

2. NLP に対する内点法

本研究では，次の (P) を NLP の標準形とする．

$$(P)\left|\begin{array}{ll}\text{minimize} & f(\boldsymbol{x})\\ \text{subject to} & \boldsymbol{g}(\boldsymbol{x})=\boldsymbol{0}, \boldsymbol{h}(\boldsymbol{x})\geq\boldsymbol{0}\end{array}\right.$$

ただし，$f:\mathbb{R}^n\to\mathbb{R}$, $\boldsymbol{g}:\mathbb{R}^n\to\mathbb{R}^l$, $\boldsymbol{h}:\mathbb{R}^n\to\mathbb{R}^m$ であり，これらはいずれも 2 回連続微分可能であるものとする．

ここで，(P) に対して次の問題を考える．

$$(P_{\mu_k})\left|\begin{array}{ll}\text{minimize} & f(\boldsymbol{x})-\mu_k\displaystyle\sum_{j=1}^{m}\log s_j\\ \text{subject to} & \boldsymbol{g}(\boldsymbol{x})=\boldsymbol{0}\\ & \boldsymbol{h}(\boldsymbol{x})-\boldsymbol{s}=\boldsymbol{0},\ \boldsymbol{s}>\boldsymbol{0}\end{array}\right.$$

ただし，$\mu_k\ (>0)$ はバリアパラメータである．$\mu_k\downarrow 0$ となるような点列 $\{\mu_k\}$ に対し，(P_{μ_k}) の解は (P) の解と漸近的に一致することが知られている．

また，(P_{μ_k}) の KKT 条件は以下のようになる．

$$\boldsymbol{r}(\boldsymbol{w};\mu_k)=\begin{bmatrix}\nabla f(\boldsymbol{x})-\nabla\boldsymbol{g}(\boldsymbol{x})\boldsymbol{y}-\nabla\boldsymbol{h}(\boldsymbol{x})\boldsymbol{z}\\ S\boldsymbol{z}-\mu_k\boldsymbol{e}\\ \boldsymbol{g}(\boldsymbol{x})\\ \boldsymbol{h}(\boldsymbol{x})-\boldsymbol{s}\end{bmatrix}=\boldsymbol{0}$$

ただし，$\boldsymbol{w}=(\boldsymbol{x},\boldsymbol{s},\boldsymbol{y},\boldsymbol{z})^\top$, $\boldsymbol{e}=(1,\ldots,1)^\top$, $S=\mathrm{diag}(s_1,\ldots,s_m)$ である．また，$\boldsymbol{y}$, $\boldsymbol{z}(>\boldsymbol{0})$ はそれぞれ等式制約，不等式制約に対する Lagrange 乗数である．

(P) を解く内点法は次のようになる．

アルゴリズム IP

Step 1: $M>0$, $\mu_0>0$ を与える．$k:=0$.

Step 2: $\|\boldsymbol{r}(\boldsymbol{w}_{k+1};\mu_k)\|\leq M\mu_k$ を満たすような $\boldsymbol{w}_{k+1}$ を求める．

Step 3: $\boldsymbol{w}_{k+1}$ が (P) の KKT 条件を十分な精度で満たしていれば，これを解として出力し，計算終了．

Step 4: μ_{k+1} を $0<\mu_{k+1}<\mu_k$ となるように選ぶ．

Step 5: $k:=k+1$ とし，Step 2 へ．

3. 実装の改良

noa_ip では，反復点の更新に Armijo のステップサイズルールを用いているが，そこでのメリット関数は，

$$\phi(\boldsymbol{x},\boldsymbol{s};\mu_k)=f(\boldsymbol{x})-\mu_k\sum_{j=1}^{m}\log s_j+Q(\boldsymbol{x},\boldsymbol{s})$$

と書くことができる．ここで $Q(\boldsymbol{x},\boldsymbol{s})$ は制約条件に対するペナルティ項であるが，これは

$$Q(\boldsymbol{x},\boldsymbol{s})=\rho\left[\sum_{i=1}^{l}|g_i(\boldsymbol{x})|+\sum_{j=1}^{m}|h_j(\boldsymbol{x})-s_j|\right]\quad(1)$$

$$Q(\boldsymbol{x},\boldsymbol{s})=\sum_{i=1}^{l}\bar{\rho}_i|g_i(\boldsymbol{x})|+\sum_{j=1}^{m}\bar{\bar{\rho}}_j|h_j(\boldsymbol{x})-s_j|\quad(2)$$

などとすることができる．(1) の ρ，(2) の $\bar{\rho}_i$, $\bar{\bar{\rho}}_j$ はそれぞれペナルティパラメータである．従来の我々の実装では (1) を用いていたが，今回の実装では (2) を用いた．これは，メリット関数において，各制約条件の違反量を Lagrange 乗数の（絶対値の）大きさに応じて反映するためのものである．

この改良により求解できる問題数が増えることが確認できたため，以下では (2) を採用している．

4. 倍精度計算と任意精度計算の比較

noa_ip を用いて，倍精度計算と任意精度計算の求解状況の比較を，求解できる問題数と計算時間の 2 点に着目して行った．計算対象は，[1] に含まれているベンチマーク問題のうちの 107 問である．

本実験における任意精度計算では，有効桁数として 10 進 1024 桁が保証されるように設定した．また，計算の終了条件は，

- 倍精度：KKT 条件の残差 $\leq$ 1e-8
- 任意精度：KKT 条件の残差 $\leq$ 1e-20

と設定した．さらに，アルゴリズム IP の反復回数の上限は 1000 回に設定した．

表 1: 実装・計算環境

OS	Ubuntu 18.04.1
CPU	Intel(R) Core(TM) i9-7980XE
Memory	128GB
Compiler	g++ 7.4.0
Library	GMP 6.1.2, MPACK 0.8.0

4.1. 求解できる問題数の比較

表 2: 求解できる問題数の比較

	求解可能	反復上限	異常終了
倍精度	85	5	17
任意精度	96	6	5

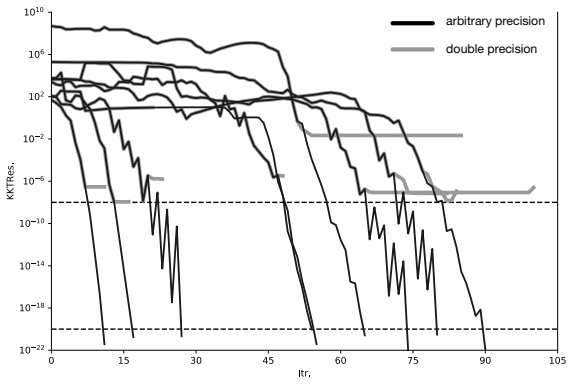

図 1: KKT 条件の残差の変化（倍精度/任意精度）

表 2 は，倍精度計算と任意精度計算で求解できる問題数を比較したものである．表 2 からわかるように，倍精度計算で求解できない問題でも任意精度計算で求解できる問題 ($\star$) がある．

図 1 は，($\star$) に該当する問題を求解する際の，KKT 条件の残差の変化を示したものである（一部抜粋）．図 1 からわかるように，倍精度計算では KKT 条件の残差が所定の値まで収束しない場合[1]であっても，任意精度計算であれば KKT 条件の残差が十分に収束していることがわかる．この結果は，我々の実装のバグが原因である可能性もあるが，倍精度計算と任意精度計算での実装は極力一致させている [3] ことから，任意精度計算の優位性を示唆する結果と判断している．

4.2. 計算時間の比較

いくつかの問題（9 題）に対し，反復点の更新 1 回に要する時間を倍精度計算/任意精度計算で比較したものが表 3 である．ここで，($\star\star$) は任意精度計算で 1 回の更新に要した時間に対する倍精度計算でのそれとの比である．

表 3 から，任意精度計算は倍精度計算に対して，数十倍から数百倍の計算時間を要することがわかる．

表 3: 反復 1 回あたりの計算時間の比

問題番号	($\star\star$)	問題番号	($\star\star$)	問題番号	($\star\star$)
015	140.26	086	85.85	109	96.90
025	269.15	106	57.71	111	244.20
084	73.83	107	70.21	116	32.73

5. おわりに

本研究では，noa_ip の実装を改良し，それを用いて任意精度計算の性能評価を行った．その結果，任意精度計算は倍精度計算と比べて高い求解精度を持つが，計算に長い時間を要することが確認された．今後の課題としては，倍精度計算から任意精度計算への切替手法の提案と実装が挙げられる．

なお本研究は JSPS 科研費 18K11185 の助成を受けたものである．

参考文献

[1] W. Hock and K. Schittkowski, Test Examples for Nonlinear Programming Codes, Lecture Notes in Economics and Mathematical Systems 187, Springer-Verlag (1981).

[2] 野口 将嗣，檀 寛成，自動微分ソフトウェアの機能追加と性能比較，日本オペレーションズリサーチ学会 2019 年春季研究発表会アブストラクト集，80-81 (2019).

[3] 崎山 皓瑛，野口 将嗣，檀 寛成，非線形最適化問題に対する任意精度計算可能な内点法の実装，日本オペレーションズリサーチ学会 2019 年春季研究発表会アブストラクト集，82-83 (2019).

[1] この場合，倍精度計算は数値エラーにより終了する．