B-027

有理数演算における 最小演算精度を用いた桁数増加の抑制の提案と評価

Reduction of Digit Number in Rational Arithmetic with Minimum Precision of Numerical Computation with Error Free

今村　翼 †　　古賀 雅伸 †
Tsubasa Imamura　　Masanobu Koga

1. はじめに

高精度な数値計算のために動的に拡張可能な多桁の自然数を分母・分子とする有理数を用いて四則演算を行う有理数演算の研究が行われている [1]。算術演算を分数に対して行えば、誤差をまったく蓄積することなしに数値計算を実行できるため高精度な数値計算が可能である。しかし、有理数演算を繰り返し行うと有理数を構成する分母・分子の桁数が膨張しやすく、桁数の膨張により計算不能に陥ることもある。通常のガウスの消去法においては計算誤差を抑制するためにピボッティングを行うが、有理数演算では計算誤差は発生しない。そこで、本研究では有理数演算によるガウスの消去法に対して桁数増加を抑制するピボッティングを提案し、その有効性を評価する。

2. 桁数増加を抑制するピボッティング

2.1 *LongEFFloat*

LongEFFloat[2] は Java 言語で記述された数値計算ライブラリであり、浮動小数点数に対して，無誤差変換 [3],[4] を用いることにより除算を除く四則演算を無誤差で行うことができる。除算においては誤差が発生しないように分子・分母を多倍長浮動小数点数とする有理数で表わすことにより除算を乗算に帰着させ無誤差で演算を行うことができる。無誤差変換を用いて，演算結果を厳密に表現するために必要な最小演算精度を求めることができ、この手法を用いることで有理数の分母・分子を表現するために必要な最小精度を求めることができる。

2.2 対角成分と桁数の小さい要素の交換

前進消去において対角成分は消去される列すべての計算に関わるため対角成分の値をその列の中で一番小さい桁数のものと交換することにより、桁数増加が抑制されることが期待できる。

Algorithm 1 桁数を考慮したピボッティング

for $k = 1$ to n **do**
　対角成分以下で最小桁数の要素を探す
　最小桁数の要素を含む行と k 行を交換
end for

ただし、提案手法によるピボッティングを Algorithm1 に示す。前進消去において消去が行われる列を k 列、行列サイズを n としている。

†九州工業大学

3. 評価

正定対称行列 A を係数行列とする連立一次方程式

$$Ax = b$$

をガウスの消去法を用いて解く問題を例とする。今回 *LongEFFloat* を用い、計算機は Intel の Corei5-7500 を搭載したパソコンを利用した。

3.1 桁数があまり増大しない行列での比較

A 行列をフランク行列 $a_{ij} = n - \max(i, j) + 1$ とする。フランク行列の対角成分以下の要素の値はすべて等しくなるためピボッティングは起こらない。①ピボッティングなし、②桁数の小さい要素と交換の 2 手法で比較を行う。

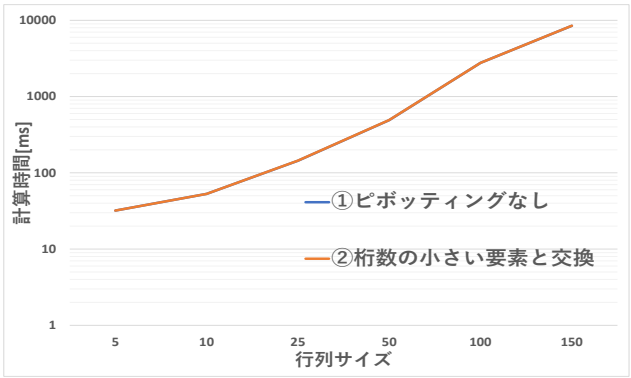

図 1: 計算時間の比較

図 1 の縦軸は計算時間 [ms]、横軸は扱ったフランク行列の行列サイズを表している。計算時間の違いは非常に小さく、ピボッティングが必要かの判断にかかるオーバーヘッドは小さい。

3.2 桁数が増大しやすい行列での比較

桁数が増大しやすい行列の例として A 行列にヒルベルト行列に手を加えた行列 $a_{ij} = \frac{1}{a*i+b*j-1}$ を用いる。今回は $a = 1$,$b = 12$,$n = 14$ とし、解ベクトルのすべての要素が 1 となるように b 行列を設定し、ガウスの消去法の前進消去、後退代入において b 行列の計算が関わる割合が大きいため、b 行列の要素の桁数の推移を見ることにより桁数の抑制を評価する。①ピボッティングなし、②対角成分と桁数の小さい要素の交換、③対角成分と絶対値の大きい要素の交換、④対角成分と桁数の大きい要素の交換の 4 手法によって桁数を比較する。図 2 は b 行

列の上から 14 番目の要素の桁数の推移を表しており、縦軸は分子・分母を表現するために必要な 10 進数での桁数、横軸は行列の要素ごとの計算回数である。

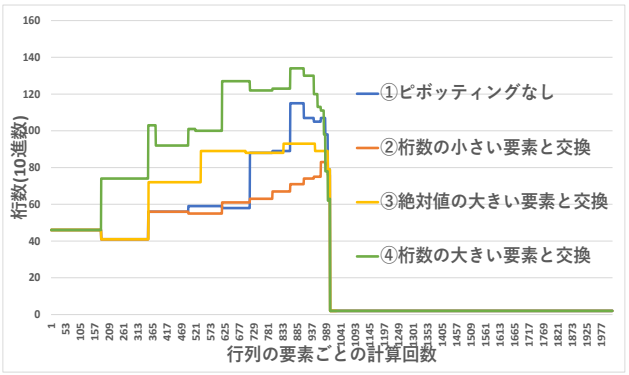

図 2: 桁数の比較 (b_{14})

さらに、b 行列の上から 1〜14 番目の要素について①, ②での抑制率比較を以下の表 1 にまとめる。今回、抑制率を

$$\text{抑制率} = \frac{\text{①での桁数増加量} - \text{②での桁数増加量}}{\text{①での桁数増加量}}$$

として設定する。b_{14} において①の手法では増加量は 46 → 115 で 69,③の手法では 46 → 83 で 37 となり抑制率は約 46 %となる。

表 1: 桁数抑制率

要素番号	抑制率 [%]	要素番号	抑制率 [%]
1	0	8	13
2	0	9	28
3	0	10	40
4	-5	11	24
5	5	12	1
6	31	13	32
7	35	14	46

表 1 より 14 個の b 行列の要素のうち、4 番目の要素の抑制率が負の値になっているため桁数が増加しているが、5〜14 番目の 10 個の要素では抑制率は正の値となっており桁数増加抑制の効果が表れていることがわかる。さらに、図 2 より対角成分と桁数の大きい要素、あるいは絶対値の大きい要素を交換することはピボッティングなしの場合と比較して有理数演算のガウスの消去法においてよいとは限らない。

4.2 節と同様の行列、手法を用いて計算時間を比較する。図 3 の縦軸は計算時間 [ms]、横軸は行列サイズを表している。

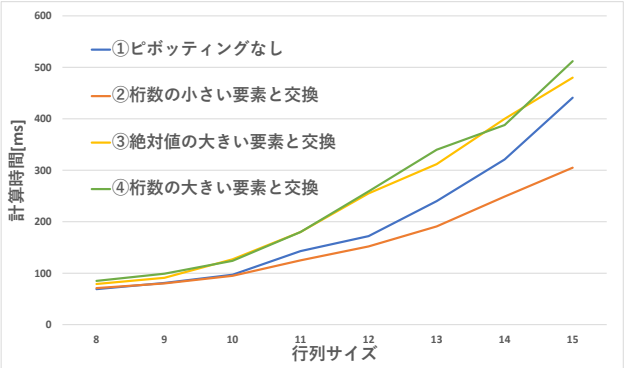

図 3: 計算時間の比較

図 3 から、提案した手法によって他の 3 手法よりも計算時間を短縮できている。そして、通常の浮動小数点数演算で用いられるピボッティングは有理数演算では計算時間に悪い影響を与える場合がある。行列サイズが大きくなるほど必要となる桁数も大きくなることが予想できるため、提案手法の効果はより増していくことが期待できる。

4. おわりに

本研究では，有理数演算における桁数増加を抑制するピボッティングを提案し評価を行った。提案手法の導入で桁数増加が抑制され計算時間を短縮することができたことから、提案手法による計算時間の短縮のほうがピボッティングを行うことによるオーバーヘッドよりも大きいことがわかる。今後はさらなる桁数増加を抑制する手法の提案をしていきたい。

参考文献

[1] 寒川光. 有理算術演算における最大公約数計算について. Technical Report 8, 早稲田大学理工学術院, apr 2016.

[2] 井川尚幸, 古賀雅伸. 無誤差演算のための最小演算精度の提案と評価. 第 14 回情報科学技術フォーラム, 2015.

[3] D.E.Knuth. The Art of Computer Programming（日本語版）. アスキー, 2004.

[4] T.J.Dekker. *A Floating-point Technique for Extending the Available Precision*. Numer. Math, 1971.