A-011

浮動小数点フォーマットに基づく事前誤差解析を用いた多倍長演算における演算精度の自動調整アルゴリズムの提案と評価

Proposal and Evaluation of Algorithm for Automatic Adjustment of Arithmetic Accuracy in Multiple-Precision Arithmetic Using Pre-error Analysis Based on Floating-Point Format

野口 剛史 † 古賀 雅伸 † 川端 悠一郎 † 矢野 健太郎 ‡
Goshi Noguchi Masanobu Koga Yuichiro Kawabata Kentaro Yano

1. はじめに

一般に数値計算では倍精度浮動小数点数型が用いられる．数学的に厳密な式が与えられたとしても，丸め誤差などが混入するため，計算機上では精度の高い結果が得られるとは限らない．

数値計算における精度には，演算精度 (precision) と結果精度 (accuracy) の 2 つの精度があり，演算精度とは，浮動小数点数の仮数部の桁数 p で，IEEE754 規格では単精度の場合 p=24，倍精度の場合 p=53 である．これに対し結果精度は，真値に対する計算結果の誤差であり，問題に依存するため演算精度だけでは分からない．また計算結果を見ても，その解の結果精度を判断することは難しい．多倍長演算を用いれば，倍精度以上の演算精度を指定して計算できるが，結果精度を判断することは難しく、演算精度をどのように設定するかという問題がある．

本研究では浮動小数点フォーマットに基づいた事前誤差解析により，演算において失われる可能性がある最大ビット数を見積もる方法と，それを用いた多倍長演算精度の自動調整アルゴリズムを提案し，例題を通して評価を行う．

2. 浮動小数点フォーマットに基づく事前誤差解析

2.1 浮動小数点フォーマット

本研究では 2 進浮動小数点フォーマットとして IEEE754-2008 規格を想定する．浮動小数点フォーマットには符号部，指数部，仮数部があり，倍精度の場合それぞれに 1 ビット，11 ビット，52 ビットが割り当てられている．倍精度浮動小数点数 x は以下のような式で表現される．

$$x = (-1)^s \times (1 + \frac{d_1}{2^1} + \frac{d_2}{2^2} + \cdots + \frac{d_{52}}{2^{52}}) \times 2^{e_x} \quad (1)$$

s は符号部を示し，e は指数部を示し，$d_1, d_2, \ldots, d_{52}$ は仮数部を示す．IEEE754 規格では，浮動小数点数を正規化するために仮数部の先頭ビットを 1 としている．

多倍長演算ではこれら浮動小数点フォーマットの指数部・仮数部の長さを設定することにより，倍精度よりも高精度な計算が可能である．

2.2 誤差が発生する原理

倍精度浮動小数点数は (1) 式のように 2^e から 2^{e-52} までの 53 ビット分の重み情報を保持している．倍精度の加算，減算，乗算では 2 つのオペランドに含まれる 106 ビットの情報を用いて演算を行い，演算結果が 53 ビットで表現できない場合は近似が行われ誤差が生じる．倍精度の $c = a + b$ において誤差が発生する概念図を図 1 に示す．ただし，a，b，c の指数部の値をそれぞれ e_a，e_b，e_c とし，$a > b$ とする． 多倍長演算では a，b の仮数部

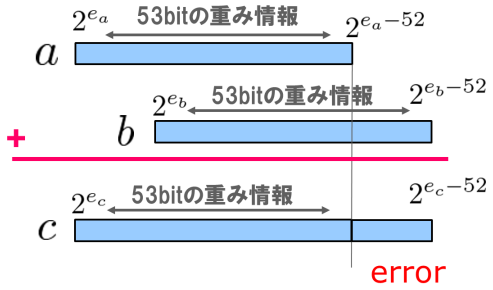

図 1: 倍精度の加算における誤差発生の概念図

が倍精度よりも長いが，倍精度の場合と同様に c で表現できない場合に誤差が生じる．

2.3 指数部・仮数部の値を考慮した事前誤差解析

2.3.1 加減算

加減算における誤差は指数部と仮数部の値にも依存している．例えば図 1 の b の値で c に入りきらずに誤差となっている部分のビットが全て 0 であるなら，c の計算において誤差が発生したことにならない．その様子を図 2 に示す．ただし，a_l，b_l は最後に出現する 1 のビットまでの長さを表している．そのため指数部と仮数部の値を調べることで，誤差ビット数の上界をより狭めることができる．

加減算における誤差ビット数の上界 err は以下のアルゴリズムで求めることができる．ただし，仮数部において一番最後に出現する 1 までのビット長をそれぞれ a_l，b_l とし，Max は引数の中で最大の値を返す関数，c の精度を p_c とする．

加減算で最大誤差ビットを求める式

$$e_{\max} = \mathrm{Max}(e_a, e_b)$$
$$l_a = a_l + (e_{\mathrm{Max}} - e_a)$$
$$l_b = b_l + (e_{\mathrm{Max}} - e_b)$$
$$\mathrm{err} = \mathrm{Max}(A_\beta, B_\beta) - p_c + 1$$

†九州工業大学
‡福岡工業短期大学

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

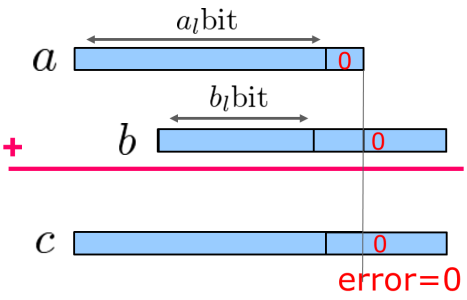

図 2: 倍精度の加算における誤差発生の概念図

2.3.2 乗算

乗算における最大誤差ビット数 err は以下のアルゴリズムで求めることができる．

乗算で最大誤差ビットを求める式

$$\mathrm{err} = a_u + b_u - c_\epsilon$$

2.4 演算精度の自動調整

多倍長変数 a，b の加算，減算，乗算は 2.3 節の方法で求められる最大誤差ビット数を用いて，誤差が全く生じないように (2) 式で演算精度を自動的に調整できる．

$$\text{新演算精度} = \text{現演算精度} + \text{最大誤差ビット数} \quad (2)$$

3. 評価

Rump の例題 [1] として知られる式

$$f = (333.75 - a^2)b^6 + a^2(11a^2b^2 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$$

$$a = 77617, b = 33096$$

を用いて，精度の自動調整の評価を行う．この問題の f の厳密な解は

$$f = -\frac{54767}{66192} = -0.8273960599\cdots$$

であるが，低精度では誤差の影響を強く受け，全く違う結果が生じる．Java 言語の多倍長クラス MPFloat[2] を利用して，自動調整を行った場合と行わない場合の演算結果を表 1 に示す．自動調整有りの場合は精度が不十分な 53bit や 106bit でも正しい結果がでていることが分かる．正しい結果が得られることが分かる．

次に Rump の例題を 1000 回解いた平均時間のグラフを表 3 に示す．計算速度については大幅に低下していることが分かる．

4. おわりに

本研究では浮動小数点フォーマットに基づいた事前誤差解析により，演算において失われる可能性がある最大

表 1: 精度の自動調整を用いた演算結果

精度 (2 進)	自動調整無し	自動調整有り
15bit	3.24519e+32	2.04003e+32
53bit	-1.18059e+21	-8.27396e-01
106bit	1.17260e+00	-8.27396e-01
159bit	-8.27396e-01	-8.27396e-01
212bit	-8.27396e-01	-8.27396e-01
265bit	-8.27396e-01	-8.27396e-01

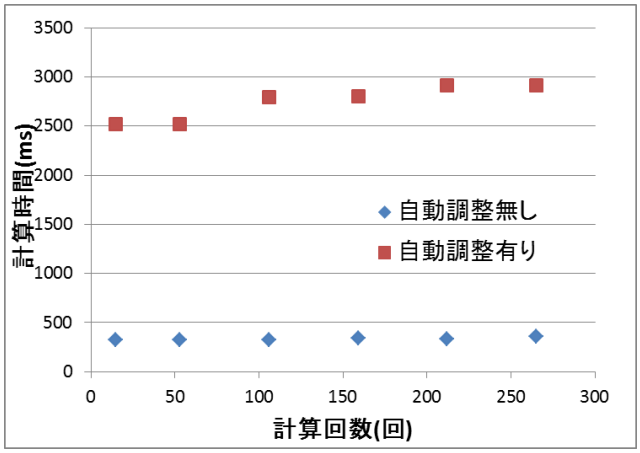

図 3: 倍精度の加算における誤差発生の概念図

ビット数を見積もる方法と，それを用いて多倍長演算精度の自動調整アルゴリズムを提案・評価した．

本研究で提案している演算精度の自動調整は，演算精度が無限に上がっていくことによるメモリ不足や速度低下が考えられるので，場合によっては最大精度を指定し，最大精度を超えたら通知もしくは演算を停止する処理が必要になると考えられる．今後の課題としては，速度の改善や，無限小数が発生しない加算，減算，乗算だけではなく，無限小数が発生する除算や sqrt などについてのアプローチや，他の精度を向上させるための手法 [3] と組み合わせることが挙げられる，

参考文献

[1] Ramon E. Moore. Reliability in computing : the role of interval methods in scientific computing. *Academic Press*, 1988.

[2] 山村英介, 古賀雅伸, 矢野健太郎, 山田賢治. 多倍長計算を用いた制御系設計パッケージ. 第 52 回システム制御情報学会研究発表講演会, 2008.

[3] 中島大雅, 古賀雅伸, 矢野健太郎. 任意精度保証計算のための多倍長演算における仮数部のビット長の適応制御の提案と評価. 情報処理学会論文誌. コンピューティングシステム, Vol. 3, No. 3, pp. 22–30, sep 2010.

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.