1J-08

任意桁数の精度保証が可能な数値計算ライブラリ IFN の区間演算ライブラリ MPFI を用いた高速化

村田 早夜香 † 川端 英之 ‡ 弘中 哲夫 ‡
広島市立大学情報科学部情報工学科 † 広島市立大学大学院情報科学研究科 ‡

1 はじめに

高精度な計算を実現する枠組みとして，我々は IFN (Improving Floating-Point Numbers) ライブラリの開発を進めている [1]. IFN ライブラリを用いた数値計算では，ユーザは精度保証方式の詳細を意識することなく，必要な精度で正確な計算結果を得ることが可能である．しかしながら，IFN ライブラリは，計算速度やメモリ使用量などの観点で改善の余地がある．

本研究では，IFN ライブラリにおいて計算結果の精度保証に用いている方式を，多倍長浮動小数点表現に基づく区間演算ライブラリ MPFI (Multiple Precision Floating-Point Interval Library) [2] を使用して再構成することにより，高速化を主眼とした改善を試みた．本稿では，2 種類の再構成方式についての実測結果を報告する．

2 実数計算ライブラリ IFN

一般的な数値計算では，IEEE754 規格に基づく浮動小数点表現による数値表現が用いられるが，倍精度であっても 10 進数 15 桁程度しか正確に表せない．一方，多倍長浮動小数点計算ライブラリの MPFR (GNU Multiple Precision Floating-Point Reliable Library)[3] などは，任意精度の仮数部長を持たせられるが，演算結果の精度保証ができない．精度保証が可能な演算方法として，例えば MPFI は，上限と下限から成る区間で 1 つの数値を表す．しかしながら，単純な区間演算は実数計算とは言えず，ユーザの所有の精度での正しい結果が得られるわけではない．

我々は研究・開発を行っている IFN ライブラリは，実数計算が可能なライブラリであり，C と Haskell から利用可能である．また，同様に正確な実数計算を実現するためのライブラリとしては，iRRAM[4] などが挙げられる．

3 IFN の挙動とその高速化の余地

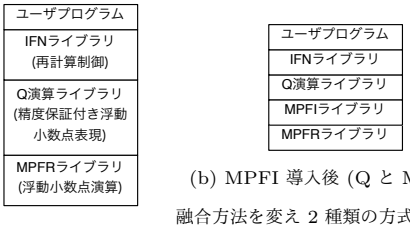
IFN は精度保証付きの浮動小数点演算で構成されている [1]．オリジナルの IFN ライブラリの階層を図 1 (a) に示す．IFN ライブラリでの計算では，ユーザが求めた精度より答えの数値の精度が低ければ，求めた精度の数値が得られるまで再計算が行われる．

オリジナルの IFN では，データ型 Q 上での Q 演算と精度保証付き浮動小数点演算を使用している [1][5]．この方式では，多倍長浮動小数点表現自体に区間に関する意味を織り込んで精度保証を実現し，Q のインスタンス $q = (s, [b_1, \cdots, b_n], e)$ が近似する実数 v の存在範囲は，$\hat{q} - \tilde{q} \leq v \leq \hat{q} + \tilde{q}$ と定義される (この時，$\hat{q} = s \times (b_1 s^{-1} + \cdots + b_n 2^{-n}) \times 2^e$，$\tilde{q} = 2^{e-n-1}$，$b_i = 0$ or 1，$s = 1$ or -1) ．精度保証のための後処理を除く浮動小数点演算には，MPFR ライブラリが用いられる． しかし，精度保証付き浮動小数点演算は時間がかかり，効率的でない部分がある．例えば，データ型 Q の加算において，MPFR ライブラリによる加算を 4 回行っている．しかしながら，その演算には，仮数部の殆どが 0 である浮動小数点同士の加算も含まれるなど，高速化のための改善の余地がある．

4 MPFI を用いた高速化の試み

本研究では，精度保証付き浮動小数点演算である Q 演算を，区間演算ライブラリ MPFI を用いて記述することにより高速化を試みる．MPFI 導入後のライブラリ階層を図 1 (b) に示す．

データ型 Q をどう実装するかによって，2 種類の方法が考えられる (表 1)．以下，それぞれについて説明する．



(a) オリジナル

(b) MPFI 導入後 (Q と MPFI の融合方法を変え 2 種類の方式を試みる)

図 1: オリジナルと MPFI 導入後のライブラリ階層

表 1: IFN ライブラリの特徴

	オリジナル	構成 1	構成 2
データ型 Q の定義	特殊な精度保証付き浮動小数点表現	実質的に MPFI	特殊な精度保証付き浮動小数点表現
Q 演算の演算方法	1.Q → MPFR 変換 2.MPFR 同士の演算 3.MPFR → Q 変換 ＋ 精度保証のための計算	MPFI 同士の演算	1.Q → MPFI 変換 2.MPFI 同士の演算 3.MPFI → Q 変換
特徴や性能	△ 計算時間に改善の余地	○ 単純な構成 ○ 型変換の必要無し △ 区間長に対して仮数部が長くなる可能性 (メモリの使用効率悪化)	○Q 演算から精度保証を省く △ 上記 1,3 の変換に時間がかかる可能性
区間同士の絶対値誤差の比較	仮数部長で比較でき容易	浮動小数点演算が必要	仮数部長で比較でき容易

4.1 MPFI を用いた実装：構成 1

構成 1 では Q を，実質的に MPFI のデータ構造を用いて表現される区間とする．ユーザは演算結果を，オリジナルとは異なり区間として得る．以下に要点を示す．

- 型変換の必要がなく，構成は単純．
- 区間の大きさに対して浮動小数点表現の仮数部が長大化してメモリの使用効率が悪い可能性．

4.2 MPFI を用いた実装：構成 2

構成 2 では，Q はオリジナルのままであり，Q 演算を MPFI で行う方法である．ユーザは演算結果を，オリジ

Performance Improvement of an Exact Real Arithmetic Library, the IFN Library, by using the MPFI Library

Sayaka Murata† Hideyuki Kawabata‡ Tetsuo Hironaka‡
†Department of Computer and Network Engineering, Hiroshima City University
‡Graduate School of Information Sciences, Hiroshima City University

表 2: IFN ライブラリにおける加乗除算をそれぞれ 10 万回行った時の演算時間の合計 [秒] と，オリジナルを基準とした時の倍率

	加算		乗算		除算	
仮数部	256	16384	256	16384	256	16384
オリジナル	0.070(1.00)	3.853(1.00)	0.102(1.00)	4.006(1.00)	0.152(1.00)	21.40(1.00)
構成 1	0.034(0.48)	0.185(0.05)	0.041(0.40)	3.433(0.86)	0.068(0.45)	6.189(0.29)
構成 2	0.147(2.09)	4.067(1.06)	0.165(1.62)	7.052(1.76)	0.197(1.29)	9.914(0.46)

ナルと同様に単一の浮動小数点表現として得ることができる．以下に要点を示す．

- Q 演算から精度保証処理を省き MPFI で実現．
- 変換の頻発による処理時間の増大が懸念される．

5 評価

5.1 評価環境

オリジナルの IFN ライブラリと本研究で作成した IFN ライブラリ (構成 1，構成 2) を用い，Q 演算での加減乗除算の計算時間と，幾つかの敏感な数値計算アプリケーションを使用して計算時間を計測した．なお，アプリケーションおよびライブラリは全て C 言語で記述されている．速度の比較を行う際の評価環境は，CPU : Intel Core i7, OS : macOS Sierra 10.12.1，メモリ : 8GB，MPFR 3.1.4, MPFI 1.5.1，C コンパイラ : Apple LLVM version 8.0.0 (-O3) である．

5.2 Q 演算での加乗除算

表 2 に示すように，構成 1 はいずれの演算においてもオリジナルより高速に演算できている．構成 2 は問題サイズが大きい時，除算がオリジナルより高速である．このことより，構成 1 は高速化が見込まれ，構成 2 はアプリケーションによっては高速化できる可能性がある．

5.3 敏感な連立一次方程式の求解

Hilbert 行列は，$h_{ij} = \frac{1}{i+j-1}$ の正方行列 (h_{ij}) で，それを係数とする連立一次方程式は正確に解くことが難しい．本実験では，ピボッティングなしの LU 分解を使用し，これの求解を行った．なお，解を絶対誤差が $\pm 2^{-129}$ 以下になるように求めた．

図 2 に示すように，問題サイズ (未知数の数) が大きくなると，構成 1 がオリジナルより高速に演算を行うことができている．一方，構成 2 はオリジナルよりも演算に時間がかかっている．また，メモリ使用量については，構成 1，構成 2 がメモリを多く使用している様子がはっきりと表れている．

5.3.1 メモリ管理

表 3 に，構成 1 において Boehm GC[6] の使用，未使用における計算時間の実測結果を示す．IFN ライブラリを使用するためには，メモリ管理の初期設定が必要であり，GC を使用する場合，IFN と Q および仮数部の領域管理をリングバッファの集合を用いて実現している．表 3 より，GC を使用した場合でも，未使用の場合に比べそれほど遅くはないことが分かり，GC を使用した場合でも有用性があると言える．

5.4 Muller の数列

Muller の数列は，$a_0 = \frac{11}{2}, a_1 = \frac{61}{11}, a_{n+1} = 111 - \frac{1130 - 3000/a_{n-1}}{a_n}$ で表される．これは正確には $\lim_{n\to\infty} a_n = 6$ となるような数列だが，通常の浮動小数点演算では桁落ちが発生し，$\lim_{n\to\infty} a_n = 100$ となる．なお，解を絶対誤差が $\pm 2^{-129}$ 以下になるように求めた．

図 3 に示すように，速度に大差はないことが分かる．したがって，このアプリケーションにおいては，オリジナルの IFN ライブラリの非効率性は低い．しかし，速度の差が開いているのは事実である．

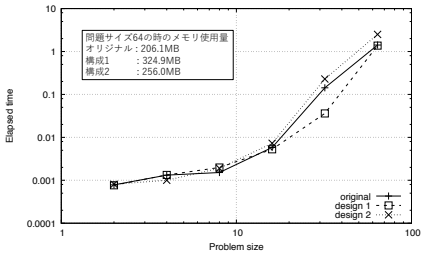


図 2: Hilbert 行列を係数とする連立一次方程式の求解 (横軸は問題サイズ，縦軸は実行に要した時間 [秒])

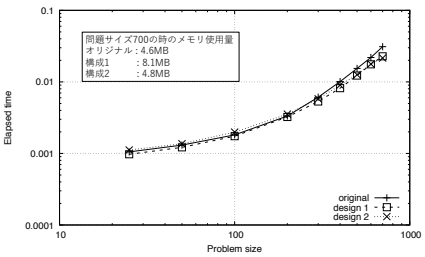


図 3: Muller の数列の計算 (横軸は問題サイズ，縦軸は実行に要した時間 [秒])

表 3: 構成 1 において GC 使用時，未使用時の演算速度 [秒] と，GC 未使用時を基準とした時の倍率 (Hilbert 行列を係数とする連立一次方程式の求解)

問題サイズ	16	64
GC 使用時	0.006679(1.3)	2.104435(1.5)
GC 未使用時	0.005264(1.0)	1.372379(1.0)

6 まとめと今後の課題

構成 1 において，Hilbert 行列を係数とする連立一次方程式の計算においては，問題サイズによってはオリジナルに比べ最大 4 倍程度高速に数値計算を行うことが可能であった．また，構成 2 はアプリケーションに依存するが，Muller の数列の計算においては，問題サイズによってはオリジナルに比べ最大 1.5 倍高速化することができた．今後，挙動の詳細な調査が必要である．

参考文献

[1] Hideyuki Kawabata and Hideya Iwasaki: Improving Floating-Point Numbers: A Lazy Approach to Adaptive Accuracy Refinement for Numerical Computations, Proc. ESOP 2016, LNCS 9632, pp.390-418, 2016.
[2] The MPFI Library, https://perso.ens-lyon.fr/nathalie.revol/software.html
[3] The GNU MPFR Library, http://www.mpfr.org/
[4] iRRAM, http://irram.uni-trier.de/
[5] 川端英之:多倍長浮動小数点演算を用いた実数計算ライブラリの実現方式, 日本応用数理学会 2016 年度年会公演論文集, 2016.
[6] The Boehm-GC Library, https://www.hboehm.info/gc/