6J-02　区間演算ライブラリ MPFI を用いた実数計算ライブラリ IFN-H の記憶領域管理を Haskell のみで記述した設計と実装

余頃花純 †　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

正確な計算を実現する Haskell ライブラリとして，我々は IFN-H (Improving Floating-Point Numbers Library for Haskell) を開発している [1]．IFN-H は，ユーザに精度保証制御の詳細を意識させることなく，任意の式に対して必要な精度での計算を行うことを可能にしている．

本研究では，IFN-H ライブラリのメモリ使用効率向上による高速化を目標とし，記憶領域管理を Haskell でのみ記述する方式での IFN-H ライブラリの設計と実装を行なった．本稿では，いくつかの実測結果について述べ，課題を整理する．

2　実数計算ライブラリ IFN-H

2.1　浮動小数点演算の問題

コンピュータを用いた実数計算は，一般的に IEEE754 規格に基づく浮動小数点表現を用いて行われる．浮動小数点演算の計算処理は高速であるが，丸め誤差や桁落ちにより精度の高い演算結果は得られるとは限らない (図 1)．さらに，ユーザが演算結果の精度を確かめることは難しい．　これら問題を解決する案として，オペランドの仮数部を長くする方法や，区間演算を用いる方法がある．多倍長浮動小数点演算ライブラリ MPFR (GNU Multiple Precision Floating-Point Reliable Library) [2] は任意の仮数部長を持つことができるが，ユーザが必要となる語調を予測することは難しい．また，区間演算ライブラリ MPFI (Multiple Precision Floating-Point Interval Library) [3] は，数値を上限と下限からなる区間で表す．しかし，ユーザが求める精度での結果が自動的に得られるわけではない．

Hilber 行列 $\boldsymbol{H}$: $h_{ij} = \frac{1}{i+j-1}$ で定義される正方行列.
$\boldsymbol{Hx} = \boldsymbol{b}, \quad \boldsymbol{b} = (1, 0, \cdots, 0)^T$
$\boldsymbol{H}$ のサイズが $n = 36$ のとき,
倍精度演算結果 ... $\tilde{x}_{36} = 6.885728522654156 \times 10^9$
正確な演算結果 ... $x_{36} = -796525272498306202 5672$

図 1: Hilbert 行列を係数とする連立一次方程式の解の誤差

2.2　IFN，及び IFN-H の概要と構成

我々は精度保証付きの実数計算を行う Haskell ライブラリ，IFN-H を開発している．IFN-H ライブラリは，多倍長浮動小数点表現とその上での演算をベースとしつつ，算術演算結果の正確さの把握機能と，要求される精度に達するまで再計算を反復する機能により，任意の数値計算プログラムに対して，ユーザの要求する精度での計算結果の導出を可能にする．

以下，IFN-H における精度保証付きの数値データ型を Q と呼び，Q 同士の演算を Q 演算と呼ぶ．Q は実数との厳密な対応関係を表した精度保証付きの浮動小数点表現で，実数の存在範囲を示している．Q 演算は，MPFR を用いて実装している．IFN-H ライブラリの階層を図 2(a) に示す．

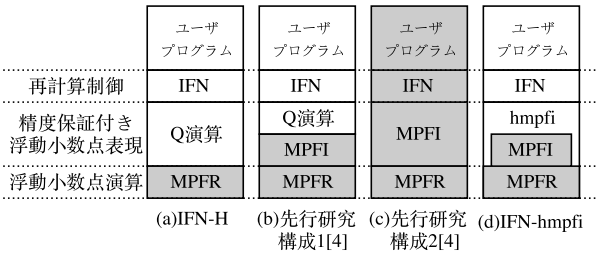

図 2: IFN ライブラリの階層
[白：Haskell ライブラリ，灰：C ライブラリ]

3　IFN-H の高速化

IFN-H の課題の一つはその高速化である．オリジナルの IFN-H は MPFR に基づく方式を取っていたが（図 2(a)）[1]，精度情報付きの数値表現である MPFI を導入することにより，C 言語版の実装（IFN-C，図 2(b)(c)）に対して高速化の検討がなされている [4]．しかしながら，遅延評価を有効活用して再計算制御を Haskell で簡潔に記述した Haskell 版（IFN-H）は C 言語部との間でのメモリ管理のための連携の実装が煩雑で，改善の余地が見られる．

本研究では，MPFI の Haskell バインディング hmpfi[5] を導入することで，Haskell ライブラリと C ライブラリのそれぞれで行われていた動的なメモリの確保と解放を，全て Haskell のメモリ管理機構が担う IFN-H ライブラリの設計と実装を行なった．hmpfi を導入した IFN-H を，IFN-hmpfi と呼ぶことにする．図 2(d) に IFN-hmpfi の階層を示す．

4　実験結果

4.1　実験環境

実験環境は次の通りである：Intel Core i7，macOS Sierra 10.12.1，メモリ 8GB，MPFR 3.1.4，MPFI 1.5.1，Apple LLVM version 8.0.0 (-O)，GHC 8.0.2.

Design and Implementation of an Exact Real Arithmetic Library, IFN-H, with the MPFI Library by using Memory Management Facility written in Haskell
Kasumi Yokoro† Hideyuki Kawabata‡ Tetsuo Hironaka‡
†Department of Computer and Network Engineering, Hiroshima City University
‡Graduate School of Information Sciences, Hiroshima City University

処理速度を比較するにあたって，C ライブラリのメモリ管理は表 1 の通り 2 つの方法を用いた．

表 1: C ライブラリのメモリ管理方法と特徴

方法 1	malloc/free で記述
方法 2	ユーザによる初期設定が必要 (MYMALLOC)

以下では，表 1 の手法に対して IFN-hmpfi がどの程度の性能であるかを比較する．

4.2 評価 1：加減乗除算の性能

表 2: 四則演算（1 万回反復）に要する実行時間（秒）

	方法 1	IFN-hmpfi
加算	0.269058	0.240104
減算	0.252423	0.237593
乗算	32.395442	31.930486
除算	0.252423	0.217424

表 1 のうち，方法 1 と IFN-hmpfi に対して，四則演算を 1 万回繰り返した結果を表 2 に示す．表 2 に示される通り，hmpfi を用いた IFN-H の実装は，四則演算の実行に要する時間の短縮には有効であるといえそうである．

4.3 評価 2：アプリケーションの処理速度

Haskell で記述された 3 つのアプリケーションの実測結果を示す．アプリケーションは図 3 に示す通りである．

表 3: 使用したアプリケーション

問題 A	Hilbert 行列を係数とする 連立一次方程式の LU 分解による求解 $h_{ij} = \frac{1}{i+j-1}$, $\boldsymbol{Hx} = \boldsymbol{LUx} = \boldsymbol{bUx} = \boldsymbol{yLy} = \boldsymbol{b}$
問題 B	Muller の数列の計算 $a_0 = \frac{11}{2}, a_1 = \frac{61}{11}, a_{n+1} = 111 - \frac{1130-3000/a_{n-1}}{a_n}$, $\lim_{n\to\infty} a_n = 6$
問題 C	Logistic 写像の計算 $x_n = ax_{n-1}(1 - x_{n-1})$

実測結果をそれぞれ図 3，図 4，図 5 に示す．横軸は問題サイズ，縦軸は実行に要した時間 （秒），図 3 の横軸は底を 2 とした対数目盛，図 4 と図 5 の横軸は底を 10 とした対数目盛である．

IFN-hmpfi の計算速度は方法 1 及び方法 2 に比べ，速度向上が達成されたとは言い難いといえる．

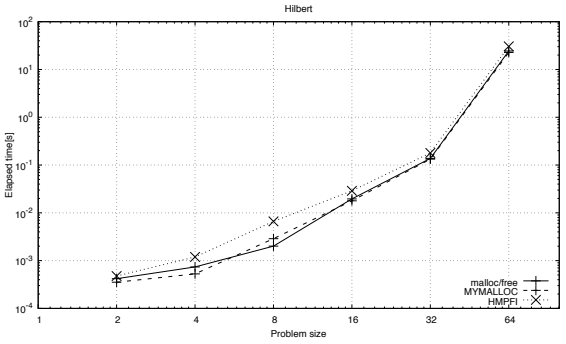

図 3: 問題 A: 線形方程式の求解

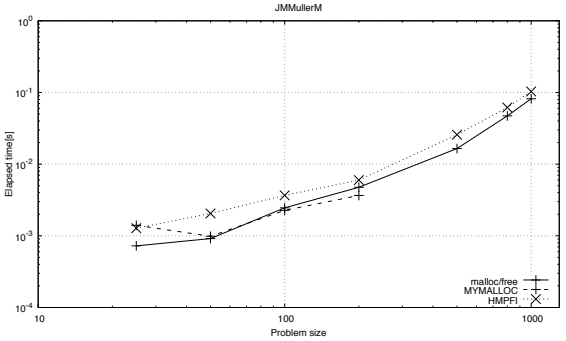

図 4: 問題 B: Muller の数列の計算

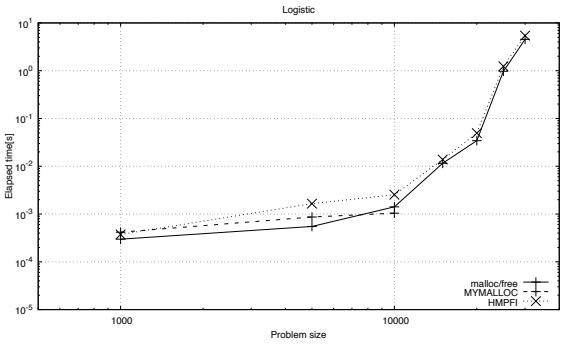

図 5: 問題 C：Logistic 写像の計算

5 まとめ

hmpfi を用いた IFN-H の実装は，個々の演算処理の高速化には有効であると見受けられるが，入り組んだ数式の計算や問題の規模が大きい場合には，既存の方法（方法 1 や方法 2）よりも高速化できるとは言えそうに無いことがわかった．IFN-H の速度向上のためには，再計算アルゴリズム自体に対する改善が必要だと考えられる．

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] MPFR, available from ⟨http://www.mpfr.org/⟩ (accessed 2018-01-12)

[3] MPFI, available from ⟨https://perso.ens-lyon.fr/nathalie.revol/software.html⟩ (accessed 2018-01-12)

[4] 村田早夜香，川端英之，弘中哲夫：任意精度を保証できる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，情報処理学会第 79 回全国大会講演論文集 (2017)

[5] 長濱ななみ，川端英之，弘中哲夫：区間演算ライブラリ MPFI への Haskell バインディングの設計と実装，情報処理学会第 79 回全国大会講演論文集 (2017)