2K-04

Rumpfi：Rustで区間演算ライブラリMPFIを手軽に扱うためのインターフェース

馬場秀人† 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

高精度な数値計算が必要な際，多倍長浮動小数点演算がよく用いられる．しかし浮動小数点演算には，丸め誤差などによる大きな誤差が計算結果に生じる場合がある．このような誤差を正確に評価できる方法に区間演算が挙げられ，区間演算ライブラリの1つにMPFI (Multiple Precision Floating-point Interval library) [1] がある．

MPFIはユーザーが指定した精度による区間演算が可能な，C言語で記述されたライブラリであり，FFIにより様々な言語で用いることができる．しかし，メモリ管理を自動で行うRust言語では，MPFIをラップして用いる際に本来ならば不要なメモリ管理の記述が必要となる．これに対して，我々はRustでMPFIを手軽に扱うためのインターフェースとしてRumpfiを開発した．本稿ではRumpfiの設計と実装について述べる．

2 区間演算ライブラリMPFI

2.1 浮動小数点演算とその誤差

コンピュータで計算を行う際は，浮動小数点演算が用いられる．浮動小数点演算とは，値を仮数部，指数部，符号部で表す浮動小数点表現 (図1) により演算を行う方法である．浮動小数点表現は仮数部長によって，値の精度が決まる．

$$-123.456789 = -1.0 \times 0.123456789 \times 10^{3}$$

符号部　仮数部　指数部

図 1: 浮動小数点表現の例

浮動小数点演算は仮数部長を大きく取ることで高精度な計算が可能であるが，桁落ちや丸め等の要因による誤差の混入を避けることはできない．誤差の混入は例え精度を高くしたとしても，本質的な改善には至らない．

2.2 MPFIの概要

浮動小数点演算における誤差を正確に評価する方法として，精度保証付き数値計算である区間演算が挙げられる．区間演算とは，上限値aと下限値bの2つの端点により表される区間 [a, b] により演算を行う方法である．

区間演算ライブラリの1つに，ユーザーが任意に値の精度を指定できるC言語で記述されたライブラリMPFIがある．MPFIでは，区間の端点に任意精度浮動小数点演算ライブラリGNU MPFR (Multiple Precision Floating-Point Reliable library) [2] の浮動小数点表現が用いられている．図2はMPFIを用いたプログラム例である．

ユーザーは7,8行目の関数`mpfi_init2()`により区間の各端点の仮数部のメモリ領域を確保し，13,14行目の関数`mpfi_clear()`で確保したメモリ領域を解放するなどして，メモリの管理をする必要がある．これによりメモリが解放済みの無効な変数の参照や，メモリの解放し忘れによるメモリリークといったヒューマンエラーが発生する恐れがある．

```c
#include <stdio.h>
#include <mpfi.h>
#include <mpfi_io.h>

int main() {
    mpfi_t a, b;
    mpfi_init2(a, 200);
    mpfi_init2(b, 200);
    mpfi_set_str(a, "1.234567", 10);
    mpfi_set_str(b, "9.876543", 10);
    mpfi_add(a, a, b);
    mpfi_out_str(stdout, 10, 0, a);
    mpfi_clear(a);
    mpfi_clear(b);
    return 0;
}
```

図 2: MPFIを用いたプログラム例

```rust
pub mod rumpfr;
pub mod rumpfi;

fn main() {
    let mut a = rumpfi::gen_ifloat(200);
    let mut b = rumpfi::gen_ifloat(200);
    rumpfi::set_str(&mut a, "1.234567");
    rumpfi::set_str(&mut b, "9.876543");
    rumpfi::add(&mut a, &a, &b);
    rumpfi::out_str(10, 0, &a);
}
```

図 3: Rumpfiを用いたプログラム例

3 プログラミング言語Rust

3.1 所有権によるメモリ管理

プログラミング言語Rustは，所有権という手法によって自動でメモリを管理する特徴を持つ．

Rustでは，プログラム中の値は基本的に1つの変数のみと対応する．この変数を所有者という．各変数には，それぞれスコープと呼ばれるその変数を参照することが可能な範囲が存在する．Rustコンパイラの制御が所有者である変数のスコープ外に移った場合，その変数はもう参照されなくなるため，その所有者の値は不要なものと判断される．その際にRustコンパイラがその値のメモリ領域を解放することで，Rustはメモリを管理している．

Rustの自動的なメモリ管理は数値計算プログラムにおいても効果的であるため，RustのMPFIインタフェースの開発は有用であると考えられる．

3.2 RustのFFI

RustでMPFIを扱う際は，外部関数インタフェースであるFFI (Foreign Funciton Interface) によりMPFIの関数を呼び出すことができる．しかし，MPFIを用いるには，ユーザーの手によるメモリ管理の記述が必要である．そのため，MPFIをFFIによってただ用いるだけでは，ユーザーがRust上でも本来なら不要な区間のデータのメモリの管理をする必要がある．

Rumpfi: An Easy-to-Use Interface from Rust to the Interval Arithmetic Library MPFI
Hideto Baba† Hideyuki Kawabata‡ Tetsuo Hironaka‡
†Department of Computer and Network Engineering, Hiroshima City University
‡Graduate School of Information Sciences, Hiroshima City University

4 Rumpfi

4.1 Rumpfi の概要

Rust で MPFI を扱う際のメモリ管理の問題に対して，我々は Rust で MPFI を手軽に扱うためのインタフェース Rumpfi を開発した．図 3 は Rumpfi のプログラム例である．Rumpfi では所有権による自動的な区間のデータのメモリ管理が行われており，図 3 のようにユーザーは区間のデータのメモリを解放する必要がない．

4.2 Rumpfi の区間のデータ構造

Rumpfi は MPFR の Rust バインディング Rumpfr[3] の設計，実装を踏襲している．Rumpfi の区間のデータ構造を図 4 に示す．Rumpfi では MPFI での区間用のデータ型に対応するデータ型 Ifloat が定義されており，Ifloat は区間の 2 つの端点を表す MPFR のデータ型の浮動小数点数を保持している．端点の仮数部のメモリは，Rust の動的配列 Vec<T> によりユーザーの指定した精度に合わせた大きさで確保される．しかし，このままでは MPFI にデータを渡す際に仮数部のメモリを保持する Vec<T> が所有権によって解放されてしまう．そこで，Rumpfi では Ifloat に新しく Vec<T> へのポインタを導入することで，Vec<T> のメモリの解放を防いでいる．Rumpfi は Ifloat を用いて MPFI の代わりに区間のデータのメモリを Rust 下で確保しており，Rust 下での所有権によるメモリ管理を実現している．

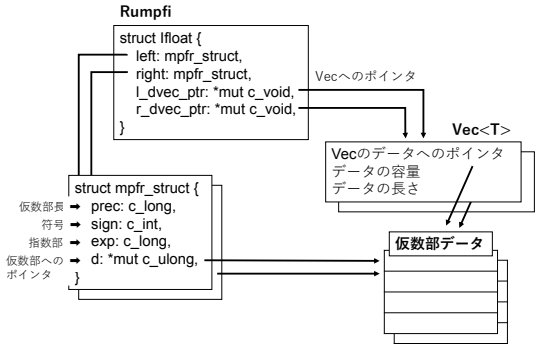

図 4: Rumpfi の区間のデータ構造

5 評価

5.1 評価環境

Rumpfi は MPFI のインタフェースとして用いられるため，Rumpfi による Rust プログラムが MPFI による C プログラムよりも処理速度に大きな遅れがないかを評価する．また，Rumpfi が Rumpfr に対してどの程度の処理速度を実現できているのか確認する．評価環境は表 1 に示す．

表 1: 評価環境

CPU	2.3GHz デュアルコア Intel Core i5
OS	macOS Monterey バージョン 12.1
メモリ	16 GB 2133 MHz LPDDR3
C コンパイラ	Apple clang version 13.0.0 (-O3)
Rust コンパイラ	rustc 1.56.1 (–release)
MPFR	MPFR 4.1.0
MPFI	MPFI 1.5.3

5.2 評価 1: 加算 10,000 回の比較

Rumpfi による単純な処理の評価として，加算 10,000 回の実行速度を比較した．結果を図 5 に示す．縦軸は底が 10 の対数目盛で処理の経過時間 (秒) を，横軸は底が 2 の対数目盛で仮数部長 (bit) を表している．

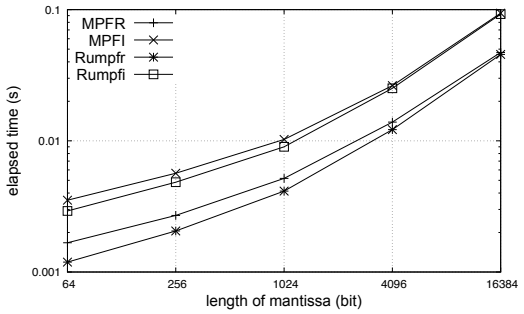

図 5: 加算 10,000 回の経過時間

5.3 評価 2: 連立一次方程式の LU 分解

Rumpfi による複雑な処理の評価として，ランダムに生成した行列を用いて連立一次方程式の LU 分解の実行速度を比較した．結果を表 2 に示す．

表 2: 連立一次方程式の LU 分解の実行速度

() 内の数値はそれぞれ MPFR と MPFI の表の値に対する比を示す

仮数部長	64 bit		4096 bit	
サイズ	100	400	100	400
MPFR	0.032 (1.00)	0.981 (1.00)	0.598 (1.00)	36.942 (1.00)
Rumpfr	0.021 (0.66)	1.036 (1.06)	0.645 (1.08)	39.439 (1.07)
MPFI	0.072 (1.00)	3.644 (1.00)	1.264 (1.00)	77.770 (1.00)
Rumpfi	0.073 (1.01)	3.782 (1.04)	1.361 (1.08)	83.810 (1.08)

図 5，表 2 のように，Rumpfi による Rust プログラムは MPFI による C プログラムとほとんど遜色ない速度で実行されていることが分かる．また，MPFI に対する Rumpfi の比較結果は，MPFR に対する Rumpfr の比較結果と同じような結果であった．よって，MPFI のインタフェースとして Rumpfi の処理速度は妥当であると言える．

6 まとめ

ユーザーによるメモリの管理が必要ない，Rust の MPFI のインタフェースとして Rumpfi を開発した．評価により，Rumpfi は C 言語での MPFI を用いた処理速度の実現が可能だと分かった．今後の課題としては，MPFI より高速な区間演算ライブラリ Arb[4] を用いた Rust インタフェースの開発が考えられる．

参考文献

[1] MPFI, available from ⟨https://perso.ens-lyon.fr/nathalie.revol/software.html⟩(accessed 2021-01-07).
[2] MPFR, available from ⟨http://www.mpfr.org⟩(accessed 2021-01-07).
[3] Tomoya Michinaka, Hideyuki Kawabata, Tetuo Hironaka, Rumpfr: A Fast and Memory Leak-free Rust Binding to the GNU MPFR Library, Journal of Information Processing, Vol.29, pp.676–684, 2021.
[4] Arb, available from ⟨https://arblib.org⟩(accessed 2022-01-07).