事前誤差評価を用いた線形計算の精度保証 –誤差解析から大規模計算まで–

森倉 悠介

早稲田大学 基幹理工学部 応用数理学科

y.morikura@aoni.waseda.jp

概要. 本講演では，線形計算における事前誤差評価を用いた精度保証付き数値計算法について紹介する．精度保証付き数値計算においては，通常，丸めモードの変更を行うことにより丸め誤差を把握する．しかし，丸めモードの変更が困難な数値計算環境も存在するので，そういった場合，事前誤差評価を用いた手法が有効である．本講演では，丸めモードを変更した手法，丸めモードを変更しない手法を合わせて紹介し，後半では講演者らが取り組んでいる連立一次方程式の精度保証付き数値計算法について紹介する．

Numerical Verification Method for linear algebra using a priori error estimation – From Error Analysis to Large-scale problems –

Yusuke MORIKURA

Department of Applied Mathematics, Waseda University

Abstract. In this lecture, we would like to introduce numerical verification methods for linear algebra by using a priori error estimation. Usually, for numerical verification methods, we obtain rounding error using the switch of the rounding modes. However, there exist numerical environments that do not support it. In such cases, it is effective in these environments that the methods use a priori error estimation. In this lecture, first, we will introduce the methods with the rounding modes, next these without the rounding modes. Finally, we would like to talk about the method for linear systems that we have been trying.

1. はじめに

本講演では，事前誤差評価を用いた線形計算の精度保証付き数値計算法について述べる．

現代の計算機環境において，高速な数値計算が行えるのは実数を浮動小数点数で近似して計算を行っているからである．そのため，数値計算には丸め誤差などの誤差が伴い，計算結果の精度において確実な保証を与えることは大変困難である．そこで，真の計算結果を区間に包含するという考えが導入され，区間同士の演算を区間演算と呼ぶ．精度保証付数値計算において，区間演算を行うためには IEEE 754 標準 [1] で定義された「方向付き

丸め」を用い丸め誤差の把握を行う．

しかし，方向付き丸めの変更が困難な計算機環境も存在する．そういった場合，浮動小数点演算における丸め誤差を数学的に見積もる計算方法：事前誤差評価が有効である．予め丸め誤差を見積もることにより，方向付き丸めの変更を行うことなく丸め誤差の把握ができる．

2 章では，本講演における記号の定義，準備を行う．3 章では，方向付き丸めを変更した場合における線形問題の精度保証付数値計算法について述べる．4 章では，方向付き丸めの変更を行わない場合における線形問題の精度保証付数値計算法について述べる．また，4 章では合わせて，無誤差変換と呼ばれる計算方法とそれらを用いた高精度計算についても述べる．最後に，5 章では，方向付き丸めの変更が困難な例として，ハイパフォーマンスコンピューティング環境における連立一次方程式の精度保証付き数値計算法について述べる．

2. 準備

本章では，本講演における記号の定義について述べる．本講演における数値計算は IEEE 754 標準 に基づくものを仮定する．集合の記号は以下に従うものとする．

定義 2.1 $\mathbb{R}$ を実数の集合とする．

定義 2.2 $\mathbb{F}$ を IEEE 754 標準に従う浮動小数点数の集合とする．

定義 2.3 $\mathbb{IR}$ を実数を要素に持つ区間の集合とする．

定義 2.4 $\mathbb{IF}$ を IEEE 754 標準に従う浮動小数点数を要素に持つ区間の集合とする．

IEEE 754 標準は，5 つ丸めの規則を定めているがその中でも以下の 3 種類の演算を用いる．方向付き丸め [*1] における浮動小数点演算の表記は以下に従うものとする．

定義 2.5 $\mathrm{fl}(\cdot)$：括弧内の演算を最近点への丸めで計算することを意味する．

定義 2.6 $\mathrm{fl}_{\nabla}(\cdot)$：括弧内の演算を下向きの丸めで計算することを意味する．

定義 2.7 $\mathrm{fl}_{\Delta}(\cdot)$：括弧内の演算を上向きの丸めで計算することを意味する．

例えば，$\mathrm{fl}_{\Delta}(a+b)$ は上向きに浮動小数点演算を行うことを表す．区間の定義は以下に従うものとする．

[*1] 方向付き丸めの変更方法については 3 章で述べる．

定義 2.8 上端・下端型の区間を

$$[\underline{x},\ \overline{x}] := \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\}$$

と表す．ただし，$\underline{x} \leq \overline{x} \in \mathbb{R}$．また，$\overline{x}$, $\underline{x}$ をそれぞれ区間の上端，下端と呼ぶ．

定義 2.9 中心・半径型の区間を

$$\langle c,\ r\rangle := \{x \in \mathbb{R} \mid c - r \leq x \leq c + r\}$$

と表す．ただし，$c,\ r\ \in \mathbb{R},\ 0 \leq r$．また，$c$, r をそれぞれ区間の中心，半径と呼ぶ．

本報告では，区間の変数は**大文字形式**を用いて表現する．

定義 2.10 区間 **a** の中心を mid(**a**)・半径を rad(**a**) と表す．

行列の大小関係を以下に定める．

定義 2.11 二つの行列 $X = (x_{ij}) \in \mathbb{R}^{n\times n},\ Y = (y_{ij}) \in \mathbb{R}^{n\times n}$（行列の第 ij 成分を x_{ij} によって $X = (x_{ij})$ と表す）の不等号は，すべての ij 成分において

$$X \leq Y \Longleftrightarrow x_{ij} \leq y_{ij},\ (i,\ j = 1,\ 2,\ \cdots, n)$$

が成り立つとする．

行列，ベクトルの絶対値を以下に定める．

定義 2.12 行列またはベクトルの絶対値をそれぞれ各成分の絶対値をとった行列またはベクトルとする．

$$|X| = (|X_{ij}|)$$

例えば，$x = (1,\ , 2,\ -3)$ において絶対値 $|x|$ は $|x| = (1,\ , 2,\ 3)$

行列に拡張した区間を以下に定義する．

定義 2.13 上端・下端型の区間行列を，

$$\mathbf{X} := [\underline{X},\ \overline{X}] = \{X \in \mathbb{R}^{m\times n} \mid \underline{X} \leq X \leq \overline{X}\}$$

と表す．ただし，$\underline{X} \leq \overline{X} \in \mathbb{R}^{m\times n}$

定義 2.14 中心・半径型の区間行列を，

$$\mathbf{X} := \langle C,\ R\rangle = \{X \in \mathbb{R}^{m\times n} \mid C - R \leq X \leq C + R\}$$

と表す．ただし，$C,\ R\ \in \mathbb{R}^{m\times n},\ \mathbf{O} \leq R$（**O** は零行列）．

3. 方向付き丸めの変更を行った行列の区間演算

本章では，方向付き丸めの変更を行った計算について述べる．線形計算における精度保証付数値計算においては，行列積の区間演算が大切になってくる．そのため，先ほど定義を行った行列の区間での計算を例に本章を進める．浮動小数点演算を用いて，厳密な区間演算を行うことは難しい．そこで，厳密な計算結果が含まれる区間の結果を得ることを目標に計算を考える．

まず簡単にスカラー区間における二項演算について記述する．

定理 1 $\mathbf{x}, \mathbf{y} \in \mathbb{IF}$ において [a]，

$$\mathbf{x} + \mathbf{y} = [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \subseteq [\mathrm{fl}_\nabla(\underline{x} + \underline{y}), \mathrm{fl}_\Delta(\overline{x} + \overline{y})],$$

$$\mathbf{x} - \mathbf{y} = [\underline{x} - \overline{y}, \overline{x} - \underline{y}] \subseteq [\mathrm{fl}_\nabla(\underline{x} - \overline{y}), \mathrm{fl}_\Delta\left(\overline{x} - \underline{y}\right)],$$

$$\begin{aligned}\mathbf{x} \cdot \mathbf{y} &= [\min(\underline{x}\underline{y},\ \overline{x}\overline{y},\ \underline{x}\overline{y},\ \overline{x}\underline{y}),\ \max(\underline{x}\underline{y},\ \overline{x}\overline{y},\ \underline{x}\overline{y},\ \overline{x}\underline{y})] \\ &\subseteq [\min(\mathrm{fl}_\nabla(\underline{x}\underline{y}),\ \mathrm{fl}_\nabla(\overline{x}\overline{y}),\ \mathrm{fl}_\nabla(\underline{x}\overline{y}),\ \mathrm{fl}_\nabla(\overline{x}\underline{y})),\ \max(\mathrm{fl}_\Delta(\underline{x}\underline{y}),\ \mathrm{fl}_\Delta(\overline{x}\overline{y}),\ \mathrm{fl}_\Delta(\underline{x}\overline{y}),\ \mathrm{fl}_\Delta(\overline{x}\underline{y}))],\end{aligned}$$

$$\begin{aligned}\mathbf{x}/\mathbf{y} &= \mathbf{x} \times [1/\overline{y},\ 1/\underline{y}],\ (0 \notin \mathbf{y}) \\ &\subseteq \mathbf{x} \times [\mathrm{fl}_\nabla(1/\overline{y}),\ \mathrm{fl}_\Delta(1/\underline{y})],\ (0 \notin \mathbf{y}).\end{aligned}$$

[a] 実際には，入力の誤差も考慮する必要があるが，本講演では浮動小数点数が与えられているものと仮定する

上記定理を拡張することで，行列区間同士の加減算も簡単に計算ができる．では，ベクトルの内積と行列積について考えてみよう．ベクトル $x,\ y \in \mathbb{F}^n$ が与えられた時の内積の区間の包含は，方向付き丸めを用いて以下のように計算できる [2].

$$\mathrm{fl}_\nabla\left(x^T y\right) \leq x^T y \leq \mathrm{fl}_\Delta\left(x^T y\right)$$

ここで，$\mathrm{fl}_\nabla\left(x^T y\right),\ \mathrm{fl}_\Delta\left(x^T y\right) \in \mathbb{F}$ となり，上向きへの丸めにおける計算 $\mathrm{fl}_\nabla\left(x^T y\right)$，下向きへの丸めにおける計算 $\mathrm{fl}_\Delta\left(x^T y\right)$ によって真の内積の結果 $x^T y$ を包含している．また，同様に，行列 $A \in \mathbb{F}^{m\times n},\ B \in \mathbb{F}^{n\times p}$ が与えられた時の行列積の区間の包含は以下のように計算できる [2].

$$\mathrm{fl}_\nabla(AB) \leq AB \leq \mathrm{fl}_\Delta(AB)$$

上向きへの丸めにおける計算 $\mathrm{fl}_\nabla(AB)$，下向きへの丸めにおける計算 $\mathrm{fl}_\Delta(AB)$ によって真の行列積の結果 $AB \in \mathbb{R}^{m\times p}$ を包含している．よって，上記は上端・下端型区間の形式を用

いて表記すると以下のようにかける．

$$x^T y \in [\mathrm{fl}_\nabla \left(x^T y\right), \mathrm{fl}_\Delta \left(x^T y\right)],$$
$$AB \in [\mathrm{fl}_\nabla (AB), \mathrm{fl}_\Delta (AB)].$$

このように，内積，行列積の場合方向付き丸めの変更を用いて，上向きの計算，下向きの計算をそれぞれ 1 回ずつ合計 2 回繰り返すことで真の結果を包含することができる．また，本方針は BLAS(Basic Linear Algebra Subprograms) などの最適化された数値計算ライブラリを用いることができるため性能を引き出しやすい．

では，実装について考えてみよう．実際に計算する際，MATLAB では方向付き丸めを以下の関数で変更することができる [*2]：

system_dependent('setround', mode)

ここで，mode は「inf：上への丸め，-inf：下への丸め，0.5：最近点への丸め（デフォルト）」のように切り替える方向付き丸めを表す．例えば，system_dependent('setround', inf)：上向き丸め，system_dependent('setround', -inf)：下向き丸めのように記述する．

以下に方向付き丸めの変更を用いた行列積の包含アルゴリズムを示す．

アルゴリズム 1 行列 $A \in \mathbb{F}^{m\times n}$，$B \in \mathbb{F}^{n\times p}$ において，方向付き丸めを用いて行列積を包含するアルゴリズム

function $[C_{down}, C_{up}]$ = mul(A, B)
    system_dependent($'$setround$'$, inf);
    $C_{up} = A \cdot B$;
    system_dependent($'$setround$'$, $-$inf);
    $C_{down} = A \cdot B$;
end

次に，行列が区間で与えられた場合の行列積 **AB** を考える．高速な区間行列積の実現のために，以下の手法を紹介する．本手法は中心・半径型の区間で計算を行うため，区間が拡大されるが実用的である．上端・下端型の区間行列が与えられた場合，以下のように上端・下端型の区間行列 $\mathbf{A} = [\underline{A}, \overline{A}] \in \mathbb{IF}^{n\times n}$ を中心・半径型の区間行列 $\langle A_m, A_r\rangle \in \mathbb{IF}^{n\times n}$ へと変形する．

$$A_m = \mathrm{fl}_\Delta \left((\overline{A} - \underline{A})/2 + \underline{A}\right),$$
$$A_r = \mathrm{fl}_\Delta (A_m - \underline{A}).$$

つぎに，行列積を以下の定理に従って計算を行う．

[*2] C99 準拠のコンパイラでは，fenv.h と fesetround を用いることで方向付き丸めの変更が可能である．

定理 2 $\mathbf{A} \in \mathbb{IR}^{m\times n}$, $\mathbf{B} \in \mathbb{IR}^{n\times p}$ に対して，$A_m := \mathrm{mid}(\mathbf{A})$，$A_r := \mathrm{rad}(\mathbf{A})$，$B_m := \mathrm{mid}(\mathbf{B})$，$B_r := \mathrm{rad}(\mathbf{B})$ とする．このとき，

$$\mathbf{A}\cdot\mathbf{B} \subseteq \langle A_m B_m, T\rangle,\ T := |A_m|B_r + A_r(|B_m| + B_r)$$

が成立する．

本定理は数値計算における丸め誤差を考慮していないため，実際には，方向付き丸めの変更を行いながら定理を計算する．また，定理においては実区間行列において説明がなされているが，実際の計算においては，浮動小数点数の区間行列が与えられているとして計算を行う．

以下に，上端・下端型の区間行列を中心・半径型に変換するアルゴリズム，区間行列同士の積を計算するアルゴリズムを示す．

アルゴリズム 2 上端・下端型区間行列 $\mathbf{A} = [\underline{A},\ \overline{A}] \in \mathbb{IF}^{m\times n}$ を中心・半径型区間行列 $\langle Am,\ Ar\rangle \in \mathbb{IF}^{m\times n}$ に変換するアルゴリズム

function $[A_m, A_r]$ = trans_midrad$(\underline{A}, \overline{A})$
    system_dependent('setround', inf);
    $A_m = (\overline{A} - \underline{A})/2 + \underline{A}$;
    $A_r = A_m - \underline{A}$;
end

アルゴリズム 3 中心・半径型区間行列 $\mathbf{A} = \langle A_m,\ A_r\rangle \in \mathbb{IF}^{m\times n}$, $\mathbf{B} = \langle B_m,\ B_r\rangle \in \mathbb{IF}^{n\times p}$ の区間行列積 を計算するアルゴリズム

function $[\underline{C}, \overline{C}]$ = interval_mul(A_m, A_r, B_m, B_r)
    system_dependent('setround', inf);
    $T = (|A_m| \cdot B_m + A_r \cdot (|B_m| + Br))$;
    $\overline{C} = A_m \cdot B_m + T$;
    system_dependent('setround', −inf);
    $\underline{C} = A_m \cdot B_m - T$;
end

上記アルゴリズムは実用性のため，中心・半径型の区間を用いて計算を行っているが，得られる区間行列は上端・下端型のものである．そのため，得られた結果を用いて次の計算を行う場合には，もう一度中心・半径型に変換して計算を行う．他にも，点行列と区間行列，区間行列と点行列の組み合わせが考えられるが，それぞれ定理 3 において半径を 0 にすることでアルゴリズム 3 と同様に計算できる．

4. 方向付き丸めの変更を必要としない計算

本章では，方向付き丸めの変更を必要としない計算についての紹介を行う．前章では，方向付き丸めを変更することにより行列の区間演算を行ったが，IEEE 754 標準に準拠した計算機環境であっても方向付き丸めの変更が困難な場合も存在する．例えば，GPU(Graphics processing unit) などがあげられる．GPU とは，通常，パソコンやワークステーションにおいて画像処理を担当するプロセッサである．近年，その処理能力を，画像処理を超えて汎用目的の計算への転用が進んでおり（GPGPU: General-purpose computing on graphics processing units），最近では，個人の計算機環境のみならず，スーパーコンピュータにも用いられている．そのため，このような環境の場合，IEEE 754 標準に準拠している計算機環境においてデフォルトの丸め方向になっている最近点丸めのみを用いた計算法が大変有効である．本章では，浮動小数点数演算における誤差解析と高精度計算について述べる．

4.1 浮動小数点数演算における誤差解析

本節では，浮動小数点数演算における誤差解析について述べる．数値計算における丸め誤差の解析については，早くから J. von Neumann 氏，H. Goldstine 氏，A. M. Turing 氏らによって考察されており，J. H. Wilkinson 氏によって体型的に発展した [4]．これらの仕事は，N. Higham 氏の書籍 [4] に，より詳しく書かれているためそちらも参照頂きたい．

ベクトルの総和と内積の誤差解析において，有名なものが [4] において述べられ，これまで広く知られてきた．近年，Rump らによってベクトルの総和と内積の誤差解析における改善が示された [6, 7].

本節では，まず，誤差解析の基本とクラシカルなベクトルと総和の誤差解析を紹介し，次に最新のベクトルと総和の誤差解析を紹介する．そして，例題として，行列積の包含を取り上げる．

以下に本節で取り扱う記号の定義を行う．

定義 4.1 **u** を相対丸め（相対精度）とする．これは，1.0 から最も小さい次の浮動小数点数との差を表す．倍精度浮動小数点数では $\mathbf{u} = 2^{-53}$ となる．

定義 4.2 **eta** を最も小さな正の浮動小数点数とする．倍精度浮動小数点数では $\mathbf{eta} = 2^{-1074}$ となる．

定義 4.3 **realmin** を正規化数の中で最も小さな正の浮動小数点数とする．倍精度浮動小数点数では $\mathbf{realmin} = 2^{-1022}$ となり，$\mathbf{eta} = 2\mathbf{u} \cdot \mathbf{realmin}$ である．

定義 4.4 実数 $a \in \mathbb{R}$ に対する Unit in the First Place (ufp) を表す関数を ufp(a) とする．

$$a \neq 0 \ \Rightarrow \mathrm{ufp}(a) := 2^{\lfloor \log_2 |a| \rfloor},\ a = 0 \Rightarrow \mathrm{ufp}(0) = 0$$

ufp は実数 a を 2 進数表記した際の先頭ビットを意味する．例えば ufp(65)=64, ufp(1.5)=1 を表す．また，浮動小数点数における ufp は 4 回の浮動小数点演算で求めることができる [6].

アルゴリズム 4 浮動小数点数における ufp の値を求めるアルゴリズム

function S = ufp(p)
　　$q = \varphi \cdot p$;　　% $\varphi = (2\mathbf{u})^{-1} + 1$
　　$S = |q - (1 - \mathbf{u})q|$;
end

定義 4.5 $r \in \mathbb{F}$ において，

- pred：与えた浮動小数点数より小さな浮動小数点数の中で最大の浮動小数点数
- succ：与えた浮動小数点数より大きな浮動小数点数の中で最小の浮動小数点数

を得る関数をそれぞれ pred，succ とし

$$\mathrm{pred}(r) := \max\{f \in \mathbb{F} : f < r\},\quad \mathrm{succ}(r) := \min\{f \in \mathbb{F} : r < f\}$$

と表す．

浮動小数点数における pred と succ を得るための関数は [5] などが提案されている．また，$a,\ b \in \mathbb{F},\ \circ \in \{+,\ -,\ \cdot,\ /\}$ において

$$\mathrm{pred}(\mathrm{fl}(a \circ b)) < a \circ b < \mathrm{succ}(\mathrm{fl}(a \circ b))$$

が成り立つ．ここで $b = 0$ のとき $\circ \neq /$ とする．また，pred 関数，succ 関数はベクトル・行列においても拡張することが出来るとする．ベクトル，行列においても pred 関数，succ 関数はそれぞれの要素の pred，succ を得る．また，ベクトル・行列同士の演算においても

$$x + y < \mathrm{succ}(\mathrm{fl}(x + y))$$

が成り立つ．この関係より，過大評価を許容すれば pred, succ のみでも包含は可能であるが，行列の計算などに用いるのは実用的ではない．

次に，浮動小数点演算における四則演算のクラシカルな誤差解析について述べる [4]. $a, b \in \mathbb{F}$ において，加減算は

$$\mathrm{fl}(a \circ b) = (a \circ b)(1 + \epsilon),\ |\epsilon| \leq \mathbf{u}, \circ \in \{+,\ -\},$$

と表される．よって，

$$|\mathrm{fl}(a \circ b) - (a \circ b)| \leq \mathbf{u}|a \circ b|.$$

$a, b \in \mathbb{F}$ において，乗除算はアンダーフローを考慮し

$$\mathrm{fl}(a \circ b) = (a \circ b)(1 + \epsilon) + \eta,\ \circ \in \{\times,\ /\},\ |\epsilon| \leq \mathbf{u},\ |\eta| \leq \mathbf{eta}/2,\ \epsilon\eta = 0,$$

と表される．よって，

$$|\mathrm{fl}(a \circ b) - (a \circ b)| \leq \mathbf{u}|a \circ b| + \mathbf{eta}/2$$

と表される．上記の誤差解析より，ベクトルの総和と内積の誤差解析が得られる．

定理 3 $x \in \mathbb{F}^n$ において，

$$n\mathbf{u} < 1 \Rightarrow |\mathrm{fl}(\sum_{i=1}^{n} x_i) - \sum_{i=1}^{n} x_i| \leq \gamma_{n-1} \sum_{i=1}^{n} |x_i|.$$

が成立する．ただし，$\gamma_n = \dfrac{n\mathbf{u}}{1 - n\mathbf{u}}$ である．

定理 4 $x, y \in \mathbb{F}^n$ において，

$$n\mathbf{u} < 1 \Rightarrow |\mathrm{fl}(x^T y) - x^T y| \leq \gamma_n |x|^T |y| + n \cdot \mathbf{eta}/2.$$

が成立する．ただし，$\gamma_n = \dfrac{n\mathbf{u}}{1 - n\mathbf{u}}$ である．

次に，近年，Siegfried M. Rump 氏や Claude Pierre Jeannerod 氏により考案されたベクトルの総和と内積の誤差解析について紹介する [7].

定理 5 $x \in \mathbb{F}^n$ において，

$$|\mathrm{fl}(\sum_{i=1}^{n} x_i) - \sum_{i=1}^{n} x_i| \leq (n-1)\mathbf{u} \sum_{i=1}^{n} |x_i|.$$

が成立する．

定理 6 $x, y \in \mathbb{F}^n$ において，

$$|\mathrm{fl}(x^T y) - x^T y| \leq n\mathbf{u}|x|^T |y| + n \cdot \mathbf{eta}/2.$$

が成立する．

これまで，$\gamma_n = \dfrac{n\mathbf{u}}{1 - n\mathbf{u}}$ を用いて上限が評価されていたが，Siegfried M. Rump 氏や Claude

Pierre Jeannerod 氏の功績により $n\mathbf{u} < 1$ といった条件が緩和された大変シンプルなものになっている．

これまでは，Wilkinson 氏らにならった誤差解析の手法であったが，次に，Rump 氏が提案した ufp を用いた誤差解析について紹介する．ufp を用いた誤差解析は浮動小数点数の先頭ビットの情報を使って誤差評価が可能なため，多くの場合，Wilkinson 氏らの誤差評価よりシャープな値が得られる．

$a, b \in \mathbb{F}$ において，加算は

$$f = (a + b) \Rightarrow f = (a + b) + \delta,\ |\delta| \leq \mathbf{u} \cdot \mathrm{ufp}(a + b) \leq \mathbf{u} \cdot \mathrm{ufp}(f)$$

と表される．よって，

$$|\mathrm{fl}(a + b) - (a + b)| \leq \min(|a|,\ |b|,\ \mathbf{u} \cdot \mathrm{ufp}(a + b)).$$

$a, b \in \mathbb{F}$ において，乗算はアンダーフローを考慮し

$$f = (a \cdot b) \Rightarrow f = (a \cdot b) + \delta + \eta,\ |\delta| \leq \mathbf{u} \cdot \mathrm{ufp}(a \cdot b) \leq \mathbf{u} \cdot \mathrm{ufp}(f),\ |\eta| \leq \mathbf{eta}/2,\ \epsilon\eta = 0$$

と表される．上記の誤差解析より，ufp を用いたベクトルの総和と内積の誤差解析が得られる．

定理 7 $x \in \mathbb{F}^n$ において，

$$|\mathrm{fl}(\sum_{i=1}^{n} x_i) - \sum_{i=1}^{n} x_i| \leq ((n-1)\mathbf{u} \cdot \mathrm{ufp}(\mathrm{fl}(\sum_{i=1}^{n} |x_i|))).$$

が成立する．また，

$$n\mathbf{u} < 1 \Rightarrow |\mathrm{fl}(\sum_{i=1}^{n} x_i) - \sum_{i=1}^{n} x_i| \leq \mathrm{fl}((n-1)(\mathbf{u} \cdot \mathrm{ufp}(\sum_{i=1}^{n} |x_i|))).$$

が成立する．

定理 8 $x, y \in \mathbb{F}^n$ において

$$(n+2)\mathbf{u} < 1 \Rightarrow |\mathrm{fl}(x^T y) - x^T y| \leq (n+2)\mathbf{u} \cdot \mathrm{ufp}(\mathrm{fl}(|x|^T |y|)) + \mathbf{realmin}.$$

が成立する．また，

$$2(n+2)\mathbf{u} < 1 \Rightarrow |\mathrm{fl}(x^T y) - x^T y| \leq \mathrm{fl}((n+2)\mathbf{u} \cdot \mathrm{ufp}(|x|^T |y|) + \mathbf{realmin}).$$

ufp を用いた総和と内積の誤差評価の特徴は，シャープに誤差が得られることに加え，上限を表現している不等式自体が浮動小数点演算 $\mathrm{fl}(\cdot)$ の形で書かれている点である．よっ

て，例えば，行列 $A \in \mathbb{F}^{m\times n}$, $B \in \mathbb{F}^{n\times p}$ が与えられた際に，方向付き丸めを用いず，行列積を中心・半径型の区間 $\langle C, R\rangle \in \mathbb{IF}^{m\times n}$ で包含を行うと，

$$C = \mathrm{fl}\,(AB)\,,\ R = \mathrm{fl}((n+2)\mathbf{u}\cdot \mathrm{ufp}(|A||B|) + \mathbf{realmin}\cdot e^T e)$$

と表され，最近点丸めのみを用いて行列積を包含することができる．ただし，$2(n+\mathbf{u}) < 1$, $e = (1,\ldots,1)^T$．このとき，必要な行列積の計算は，$\mathrm{fl}\,(AB)$, $\mathrm{fl}\,(|A||B|)$ の 2 回である．これは，浮動小数点演算を行ったその近似値 C とその誤差半径が R として，真の計算結果を包含している．

つぎに，中心・半径型の区間行列 $A \in \mathbb{IF}^{m\times n}$, $B \in \mathbb{IF}^{n\times p}$ が与えられた場合の行列積の包含を考える．定理 3 より $2(n+\mathbf{u}) < 1$ を仮定して中心 $A_m B_m$ と誤差半径 T の計算を考える．まず，中心 $A_m B_m$ は定理 8 より，

$$C = \mathrm{fl}\,(A_m B_m)\,,\ R = \mathrm{fl}((n+2)\mathbf{u}\cdot \mathrm{ufp}(|A_m||B_m|) + \mathbf{realmin}\cdot e^T e)$$

のように計算される．次に，$|A_m|B_r$ の上限を考える．定理 8 より

$$\begin{aligned} T_1 &:= \mathrm{fl}(|A_m|B_r),\\ ||A_m|B_r - \mathrm{fl}(|A_m|B_r)| &\le \mathrm{fl}((n+2)\mathbf{u}\cdot \mathrm{ufp}(T_1) + \mathbf{realmin}\cdot e^T e) =: T_2,\\ |A_m|B_r &\le T_1 + T_2. \end{aligned}$$

$(|B_m| + B_r)$ は

$$|B_m| + B_r \le \mathrm{succ}(\mathrm{fl}(|B_m| + B_r)) =: T_3$$

となる．次に，$A_r T_3$ は定理 8 より

$$\begin{aligned} T_4 &:= \mathrm{fl}(A_r T_3),\\ |A_r T_3 - \mathrm{fl}(A_r T_3)| &\le \mathrm{fl}((n+2)\mathbf{u}\cdot \mathrm{ufp}(T_4) + \mathbf{realmin}\cdot e^T e) =: T_5,\\ A_r T_3 &\le T_4 + T_5, \end{aligned}$$

となる．よって定理 3 における半径 T の上限は

$$\begin{aligned} T &\le T_1 + T_2 + T_4 + T_5 + R\\ &\le \mathrm{fl}(T_1 + T_2 + T_4 + T_5 + R) + \mathrm{fl}(4\mathbf{u}(T_1 + T_2 + T_4 + T_5 + R))\\ &\le \mathrm{succ}(\mathrm{fl}(T_1 + T_2 + T_4 + T_5 + R) + \mathrm{fl}(4\mathbf{u}(T_1 + T_2 + T_4 + T_5 + R))) =: T_6 \end{aligned}$$

と表される．よって，

$$\mathbf{A}\cdot\mathbf{B} \subseteq \langle C,\ T_6\rangle$$

が得られる．

アルゴリズム 5 中心・半径型区間行列 $\mathbf{A} = \langle A_m, A_r \rangle \in \mathbb{IF}^{m\times n}$, $\mathbf{B} = \langle B_m, B_r \rangle \in \mathbb{IF}^{n\times p}$ の区間行列積を方向付き丸めを用いず計算するアルゴリズム

function $[C, R]$ = interval_mul2(A_m, A_r, B_m, B_r)
  if $2(\mathrm{n}+2)\mathbf{u} < 1, \mathrm{error}('\mathrm{failed}'),$ end
  $C = A_m \cdot B_m$;
  $R = ((n+2)\mathbf{u} \cdot \mathrm{ufp}(|A_m| * |B_m|) + \mathbf{realmin} \cdot e^T e)$
  $T_1 = |A_m| \cdot B_r$
  $T_2 = ((n+2)\mathbf{u} \cdot \mathrm{ufp}(T_1) + \mathbf{realmin} \cdot e^T e)$;
  $T_3 = |B_m| + B_r$;
  $T_4 = A_r \cdot T_3$;
  $T_5 = ((n+2)\mathbf{u} \cdot \mathrm{ufp}(T_4) + \mathbf{realmin} \cdot e^T e)$;
  $R2 = (T_1 + T_2 + T_4 + T_5 + R)$;
  $T_6 = succ(R2 + 4\mathbf{u} \cdot \mathrm{ufp}(R2))$;
end

このとき，行列積を行う箇所は，

$$C:\ \mathrm{fl}(A_m B_m), \quad R:\ \mathrm{fl}(|A_m||B_m|), \quad T_1, T_2:\ \mathrm{fl}(|A_m|B_r), \quad T_4, T_5:\ \mathrm{fl}(|A_r|T_3).$$

であるが，T_1, T_2 と T_4, T_5 に現れる行列積は，全て非負行列同士の積であるため，それぞれで同じ行列積の結果が使える．そのため全体の行列積の回数は 4 回である．このように，方向付き丸めを変更することなく，区間演算が可能である．ただし，本手法は，誤差解析を行い丸め誤差を計算したため，生じうる丸め誤差の最大を見積もって計算を行っている．よって，方向付き丸めを変更した手法と比較すると区間幅の拡大が顕著にみられる．

4.2 無誤差変換を用いた高精度計算

本節では，無誤差変換と呼ばれるアルゴリズムを用いた高精度計算について簡単に述べる．前説では，事前誤差評価を用いることで方向付き丸めの変更を必要としない計算法について述べた．しかし，事前誤差評価を用いるため結果が過大評価になるそのため，高精度計算を導入し丸め誤差の増大を抑制する．

加算と乗算において，その近似値 $x \in \mathbb{F}$ とその誤差 $y \in \mathbb{F}$ に無誤差で変換するアルゴリズムが知られている．

アルゴリズム 6 $a, b \in \mathbb{F}$ の加算における無誤差変換

function $[x, y]$ = TwoSum(a, b)
  $x = (a + b)$;
  $z = (x - a)$;

$y = ((a - (x - z)) + (b - z));$
end

アルゴリズム 7 $a,\ b \in \mathbb{F}$ の乗算における無誤差変換

function $[x, y]$ = TwoProduct(a, b)
$x = (a \cdot b);$
$[a_1, a_2]$ = Split(a);
$[b_1, b_2]$ = Split(b);
$y = (a_1 \cdot b_2 - (((x - a_1 \cdot b_1) - a_2 \cdot b_1) - a_1 \cdot b_2));$
end

アルゴリズム 8 $a \in \mathbb{F}$ を $a = ah + al,\ ah,\ al \in \mathbb{F}$ に無誤差で分割するアルゴリズム．

function $[ah, al]$ = Split(a)
$c = (\text{factor} \cdot a);$ %factor $= 2^s + 1$
$ah = (c - (c - a));$ %For binary64, $s = 27$
$al = (a - ah);$
end

アルゴリズム 9 FMA[*3] を用いた $a,\ b \in \mathbb{F}$ の乗算における無誤差変換

function $[x, y]$ = TwoProductFMA(a, b)
$x = (a \cdot b);$
y = FMA$(a,\ b,\ -x)$;
end

浮動小数点演算を行うと丸め誤差の影響により，得られた近似値が真の結果と一致するとは限らないが上記アルゴリズムを用いることにより，その近似値とこれまで切り捨てていた誤差を把握することができる．これらを組み合わせることによる高精度なさまざまな計算が考案されている [10, 11]．ここでは，高精度な内積アルゴリズム [10] を紹介する．

定理 9 以下の高精度な内積アルゴリズム Dot2 [10] を実行すると

$$|\text{res} - x^T y| \le \mathbf{u}|x^T y| + \gamma_n^2 |x^T||y| + 5n\mathbf{eta}$$

という誤差評価が成立する．

[*3] $a, b, c \in \mathbb{F}$ において $a \cdot b + c$ の計算を一度に行う演算．近年の CPU や GPU に実装されている場合が多い．TwoProduct に用いることで演算回数を大幅に節約できる．

アルゴリズム 10 $x,\ y \in \mathbb{F}^n$ において計算精度の約 2 倍（倍精度浮動小数点数であれば約 4 倍）精度で計算するアルゴリズム

function [res] = Dot2(x, y)
  if $2n\mathbf{u} \geq 1$, error('inclusionfailed'), end;
  $[p, s]$ = TwoProduct($x_1,\ y_1$);
  for $i = 2 : n$
    $[h, r]$ = TwoProduct($x_i,\ y_i$);
    $[p, q]$ = TwoSum(p, h);
    $t = (q + r)$;
    $s = (s + t)$;
  end
  res $= (p + s)$;
end

また，Dot2 に誤差上限の計算を加えたアルゴリズムは以下の Dot2Err である．

アルゴリズム 11 $x,\ y \in \mathbb{F}^n$ において計算精度の約 2 倍（倍精度浮動小数点数であれば約 4 倍）精度で計算し，その誤差上限を求めるアルゴリズム

function [res, err] = Dot2Err(x, y)
  if $2n\mathbf{u} \geq 1$, error('inclusionfailed'), end;
  $[p, s]$ = TwoProduct($x_1,\ y_1$);
  $e = |s|$;
  for $i = 2 : n$
    $[h, r]$ = TwoProduct($x_i,\ y_i$);
    $[p, q]$ = TwoSum(p, h);
    $t = (q + r)$;
    $s = (s + t)$;
    $e = (e + |t|)$;
  end
  res $= (p + s)$;
  $\delta = (n\mathbf{u}/(1 - 2n\mathbf{u}))$;
  $\alpha = (\mathbf{u}|\text{res}| + (\delta \cdot e + 3\mathbf{eta}/\mathbf{u}))$;
  err $= (\alpha/(1 - 2\mathbf{u}))$;
end

次に，高精度な行列積計算法 [12] を紹介する．

アルゴリズム 12 $A \in \mathbb{F}^{m\times n}$ を $A^{(1)} + A^{(2)}$ へ丸め誤差なしに分割するアルゴリズム．

function $[A^{(1)}, A^{(2)}]$ = Split_A(A)
n = size(A, 2);
μ = max(abs(A), [ ], 2); $\quad \%\mu \in \mathbb{F}^{m\times 1}$ with $\mu(i) = \max\limits_{1\le j\le n} |a_{ij}|$
τ = 2.^ceil((53 + log2(n + 1))/2);
t_A = 2.^ceil(log2(μ)) $\cdot\ \tau$;
S_A = repmat(t_A, 1, n); $\quad \%S_A = t_A \cdot e^T$ for $e = (1, 1, ..., 1)^T \in \mathbb{F}^n$
$A^{(1)} = (A + S_A) - S_A$;
$A^{(2)} = A - A^{(1)}$;
end

アルゴリズム 13 行列 $B \in \mathbb{F}^{m\times n}$ を $B^{(1)} + B^{(2)}$ へ丸め誤差なしに分割するアルゴリズム．

function $[B^{(1)}, B^{(2)}]$ = Split_B(B)
n = size(B, 1); $\quad \%\mu \in \mathbb{F}^{1\times m}$ with $\mu(j) = \max\limits_{1\le i\le n} |A_{ij}|$
μ = max(abs(B), [ ], 1);
τ = 2.^ceil((53 + log2(n + 1))/2);
t_B = 2.^ceil(log2(μ)) $\cdot\ \tau$;
S_B = repmat(t_B, n, 1); $\quad \%S_B = e \cdot t_B$ for $e = (1, 1, ..., 1)^T \in \mathbb{F}^n$
$B^{(1)} = (B + S_B) - S_B$;
$B^{(2)} = B - B^{(1)}$;
end

上記定理は，行列 A と行列 B を以下のように分解する．

$$A = A^{(1)} + A^{(2)} + A^{(3)} + \cdots + A^{(p)},\ B = B^{(1)} + B^{(2)} + B^{(3)} + \cdots + B^{(q)}$$

このとき，行列のインデックスが若いほうが高いビットをもつように分解されている．分解された行列 A, B より，行列積 AB は

$$\begin{aligned} AB &= (A^{(1)} + A^{(2)} + A^{(3)} + \cdots + A^{(p)})(B^{(1)} + B^{(2)} + B^{(3)} + \cdots + B^{(q)}) \\ &= A^{(1)}B^{(1)} + A^{(1)}B^{(2)} + A^{(2)}B^{(1)} + \cdots + A^{(p)}B^{(q)} \end{aligned}$$

とかける．このときのポイントは各行列積

$$\mathrm{fl}\left(A^{(i)}B^{(j)}\right) = A^{(i)}B^{(j)},\ 1 \le i \le p,\ 1 \le j \le q$$

を浮動小数点演算で計算するとき誤差が生じないように分割している．そのため，行列積 AB は $C_1 = \mathrm{fl}\left(A^{(1)}B^{(1)}\right)$, $C_2 = \mathrm{fl}\left(A^{(1)}B^{(2)}\right)$, $\cdots$, $\mathrm{fl}\left(A^{(p)}B^{(q)}\right) = C_{pq}$ とおくと

$$\begin{aligned} AB &= \mathrm{fl}\left(A^{(1)}B^{(1)}\right) + \mathrm{fl}\left(A^{(1)}B^{(2)}\right) + \mathrm{fl}\left(A^{(2)}B^{(1)}\right) + \cdots + \mathrm{fl}\left(A^{(p)}B^{(q)}\right) \\ &= C_1 + C_2 + C_3 + \cdots + C_{pq} \end{aligned}$$

と変形され，行列同士の和で表される．このとき，行列積の回数は pq 回である．この和を所望の結果精度が得られるアルゴリズムなどで計算することにより高精度な結果を得ることができる．この計算方法は，最適化された BLAS などの数値計算ライブラリを用いることができるため，計算量は増大するがパフォーマンスが得られやすい．

5. 連立一次方程式の解法

本章では，方向付き丸めの変更を用いない最近点丸めのみを用いた連立一次方程式の精度保証付き数値計算とその実装についてポイントのみ簡単に述べる．連立一次方程式 $Ax = b$, $A \in \mathbb{F}^{n\times n}$, $b \in \mathbb{F}^n$ における精度保証式は以下の定理 [2] を用いる．

定理 10 連立一次方程式の近似解 $\tilde{x}$ と A の近似逆行列 R が求められたとする．

$$\|RA - I\| < 1$$

を満たすとき，

$$\|\tilde{x} - A^{-1}b\| \leq \frac{\|R(A\tilde{x} - b)\|}{1 - \|RA - I\|}$$

が成立する．ただし，I を単位行列とする．

本定理はバナッハの縮小写像原理から導かれる．本定理を計算する際は，計算の容易な最大値ノルムがよく用いられる．

では，方向付き丸めを変更し定理 10 の $\|RA - I\|$ の上限を求める方法について簡単に述べる．

- $C := \mathrm{fl}_\nabla(RA - I) \leq RA - I \leq \mathrm{fl}_\Delta(RA - I) =: D$.
- $\mathrm{fl}_\Delta(\|\max(|C|, |D|)\|_\infty)$ を計算し 1 との大小関係を確認する．

このように方向付き丸めを変更することで，簡単に $\|RA - I\|$ の上限を計算することができる．$\|R(A\tilde{x} - b)\|$ の上限についても同様に計算できるため，ここでは省略する．次に，方向付き丸めを変更せず，事前誤差評価を行った定理を示す．

定理 11 連立一次方程式 $Ax = b$, $A \in \mathbb{F}^{n\times n}$, $b \in \mathbb{F}^n$ において，A の近似逆行列 R が求められたとするとき，定理 10 における $\|RA - I\|_\infty$ の上限は，方向付き丸めの変更を用いず

$$a_1 := \mathrm{muls}(|A|),\ a_2 := \mathrm{muld}(|R|,\ a_1),$$
$$\alpha_1 := \mathrm{muls}(\mathrm{fl}(RA - I)),\quad \alpha_2 := \mathrm{succ}(\mathrm{fl}((n\mathbf{u}) \cdot c_2)),\quad \alpha_3 := \mathrm{fl}(n\mathbf{eta} \cdot e)$$

$\max(\alpha_1) < 1$ であれば

$$\|RA - I\|_\infty \le \|\mathrm{fl}(\mathrm{succ}(\alpha_1 + \alpha_2 + \alpha_3 + \mathbf{u}e) + 3\mathbf{u} \cdot \mathrm{ufp}(\alpha_1 + \alpha_2 + \alpha_3 + \mathbf{u}e))\|_\infty.$$

により得られる．ただし $r \in \mathbb{F}$, $\mathrm{succ}(r) := \min\{f \in \mathbb{F} : r < f\}$, $\mathrm{ufp}(r) := 2^{\lfloor \log_2 |r| \rfloor}$,

$$\mathrm{muls}(A) := \mathrm{succ}(\mathrm{fl}(|A|e + ((n-1)\mathbf{u} \cdot \mathrm{ufp}(|A|e)))) \ge |A|e,$$
$$\mathrm{muld}(A, x) := \mathrm{succ}(\mathrm{fl}(|Ax| + ((n+2)\mathbf{u} \cdot \mathrm{ufp}(|A||x|) + \mathbf{realmin} \cdot e))) \ge |Ax|.$$

定理 12 連立一次方程式 $Ax = b$, $A \in \mathbb{F}^{n\times n}$, $b \in \mathbb{F}^n$ において，A の近似逆行列 R，近似逆行列 $\tilde{x}$ が求められたとするとき，定理 10 における $\|R(A\tilde{x} - b)\|_\infty$ の上限は，方向付き丸めの変更を用いず

$$\mathrm{mid} = \mathrm{fl}(A\tilde{x} - b),\ \mathrm{rad} = \mathrm{fl}((n+3)\mathbf{u} \cdot \mathrm{ufp}(|A||\tilde{x}| + |b|) + \mathbf{realmin} \cdot e).$$

で残差の包含が得られ，

$$\mathrm{mid} - \mathrm{rad} \le A\tilde{x} - b \le \mathrm{mid} + \mathrm{rad},$$
$$|R(A\tilde{x} - b)| \le (|R\mathrm{mid}| + |R|\mathrm{rad}),$$
$$b_1 := \mathrm{muld}(R, \mathrm{mid}),\ b_2 := \mathrm{muld}(|R|, \mathrm{rad}),$$
$$\|R(A\tilde{x} - b)\|_\infty \le \max(\mathrm{succ}(b_1 + b_2)),$$

により得られる．ただし $r \in \mathbb{F}$, $\mathrm{succ}(r) := \min\{f \in \mathbb{F} : r < f\}$, $\mathrm{ufp}(r) := 2^{\lfloor \log_2 |r| \rfloor}$,

$$\mathrm{muls}(A) := \mathrm{succ}(\mathrm{fl}(|A|e + ((n-1)\mathbf{u} \cdot \mathrm{ufp}(|A|e)))) \ge |A|e,$$
$$\mathrm{muld}(A, x) := \mathrm{succ}(\mathrm{fl}(|Ax| + ((n+2)\mathbf{u} \cdot \mathrm{ufp}(|A||x|) + \mathbf{realmin} \cdot e))) \ge |Ax|.$$

ただし，本手法は事前誤差解析による丸め誤差の過大評価のため，大規模計算を行う際には，

- 事前誤差解析による丸め誤差の抑制：高精度な行列積の利用
- 近似解の改善：高精度内積計算アルゴリズムの利用
- 残差のシャープな評価：高精度内積計算アルゴリズムの利用

が有効である．こちら詳細については講演時紹介する．

謝辞 本講演を行う機会を頂いた早稲田大学 大石進一 教授，東京女子大学 荻田武史 准教授，芝浦工業大学 尾崎克久 准教授に心から感謝致します．また，本稿を書き上げるにあたってご助言頂きました早稲田大学 柏木雅英 教授にもこの場を借りて感謝致します．また，公私ともにお世話になっております研究室の皆様にも感謝致します．

本稿の研究の一部は，JSPS 科研費 15K15939 の助成を受けた．

参考文献

[1] ANSI/IEEE 754-1985, IEEE Standard for Binary Floating-Point Arithmetic. New York, 1985.

[2] S. Oishi and S. M. Rump, *Fast Verification of Solutions of Matrix Equations*, Numer. Math., vol. 90, no. 4, pp. 755–773, February 2002.

[3] J. H. Wilkinson, *Error analysis of floating-point computation*, Numer. Math., vol. 2, no. 1, pp 319–340, 1960.

[4] N. Higham, *Accuracy and stability of numerical algorithms, Second Edition.* SIAM Publications, Philadelphia, 2002.

[5] S.M. Rump, P. Zimmermann, S. Boldo, and G. Melquiond, *Computing predecessor and successor in rounding to nearest*, BIT Numerical Mathematics, vol. 49, no. 2, pp. 419–431, 2009.

[6] S.M. Rump, *Error estimation of floating-point summation and dot product*, BIT Numerical Mathematics, vol. 52, no. 1, pp. 201–220, 2012.

[7] C.-P. Jeannerod and S.M. Rump, *Improved error bounds for inner products in floating-point arithmetic*, SIAM. J. Matrix Anal. & Appl. (SIMAX), vol. 34, no. 2, pp. 338–344, 2013.

[8] D. E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, vol. 2, AddisonWesley, Reading, MA, 1969.

[9] T. J. Dekker, *A floating-point technique for extending the available precision*, Numer. Math., vol. 18, no. 3, pp. 224–242, 1971.

[10] T. Ogita, S. M. Rump, S. Oishi, *Accurate sum and dot product*, SIAM Journal on Scientific Computing, vol. 26, no. 6, pp. 1955–1988, 2005.

[11] T. Ogita, S. M. Rump, S. Oishi, *Accurate sum and dot product*, SIAM Journal on Scientific Computing, vol. 26, no. 6, pp. 1955–1988, 2005.

[12] K. Ozaki, T. Ogita, S. Oishi, S. M. Rump, *Error-Free Transformation of Matrix Multiplication by Using Fast Routines of Matrix Multiplication and its Applications*, Numerical

Algorithms, vol. 59, no. 1, pp. 95–118, 2012.

[13] Y. Morikura, K. Ozaki, and S. Oishi, *Verification methods for linear systems using ufp estimation with rounding-to-nearest*, Nonlinear Theory and its Applications, vol. 4, no. 1, pp. 12–22, 2013.

[14] 大石 進一, 荻田 武史, 太田 貴久: 高精度内積計算アルゴリズムを用いた連立一次方程式の精度保証付き数値計算法, 日本シミュレーション学会論文誌, 25:5 (2006), 170-178. vol. 25, no. 5, pp. 170–178, 2006.