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

三部会連携「応用数理セミナー」: 日本応用数理学会，東京大学

2015 年 12 月 24 日

はじめに

- 例えば，連立一次方程式 $Ax = b$, $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $\tilde{x}$ は近似解，の精度保証付き数値計算：

定理

R : A の近似逆行列
I : 単位行列

正則性の検証： $\|RA - I\|_\infty < 1,$

A^{-1} が存在し

誤差上限： $\|x - \tilde{x}\|_\infty \leq \dfrac{\|R(A\tilde{x} - b)\|_\infty}{1 - \|RA - I\|_\infty}.$

- この不等式を丸め誤差を考慮しながら計算したい.
- 数値線形代数計算における精度保証付き数値計算においては，行列の区間演算が必要になってくる.

本日の講演

1. はじめに
2. 準備
3. 方向付き丸めの変更を行った行列の区間演算
4. 方向付き丸めを用いない計算
5. 連立一次方程式の精度保証付き数値計算法
6. まとめ

準備：集合

定義

$\mathbb{R}$ を実数の集合とする.

定義

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

定義

$\mathbb{IR}$ を実数を要素に持つ区間の集合とする.

定義

$\mathbb{IF}$ 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.

準備：方向付き丸め

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

$\mathrm{fl}(\cdot)$：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

$\mathrm{fl}_{\bigtriangledown}(\cdot)$：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

$\mathrm{fl}_{\bigtriangleup}(\cdot)$：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．

準備：区間

定義

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

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

区間 $\mathbf{a}$ の中心を $\mathrm{mid}(\mathbf{a})$・半径を $\mathrm{rad}(\mathbf{a})$ と表す．

準備：中心半径型の区間演算

$\mathbf{x} = \langle x_c,\ x_r \rangle \in \mathbb{IR}, \mathbf{y} = \langle y_c,\ y_r \rangle \in \mathbb{IR}$ において

$$
\begin{aligned}
\mathbf{x} + \mathbf{y} &= \langle x_c + y_c,\ x_r + y_r \rangle, \\
\mathbf{x} - \mathbf{y} &= \langle x_c - y_c,\ x_r + y_r \rangle, \\
\mathbf{x} \cdot \mathbf{y} &\subseteq \langle x_c y_c,\ |x_c| y_r + |y_c| x_r + x_r y_r \rangle, \\
\mathbf{x}/\mathbf{y} &= \frac{\mathbf{x} \cdot \underline{\mathbf{y}}}{\underline{\mathbf{y}} \cdot \underline{\mathbf{y}}} = \frac{\mathbf{x} \cdot \underline{\mathbf{y}}}{y_c^2 - y_r^2} \quad (0 \notin \mathbf{y})
\end{aligned}
$$

準備：大小関係，絶対値

行列の大小関係を以下に定める．

定義

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

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 $|x| = (1,\ 2,\ 3)$

準備：行列の区間

行列に拡張した上端・下端型区間を以下に定義する.

定義

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，$\underline{X} \leq \overline{X} \in \mathbb{R}^{m\times n}$

$\mathbf{A} \in \mathbb{IR}^{2\times 2}$,

$$\mathbf{A} = \begin{pmatrix} [1,2] & [-1,1] \\ [0,1] & [-1,2] \end{pmatrix}$$

準備：行列の区間

行列に拡張した中心・半径型区間を以下に定義する.

定義

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

$\mathbf{A} \in \mathbb{IR}^{2 \times 2}$,

$$\mathbf{A} = \begin{pmatrix} \langle 1.5, 0.5 \rangle & \langle 0, 1 \rangle \\ \langle 0.5, 0.5 \rangle & \langle 0.5, 1.5 \rangle \end{pmatrix}$$

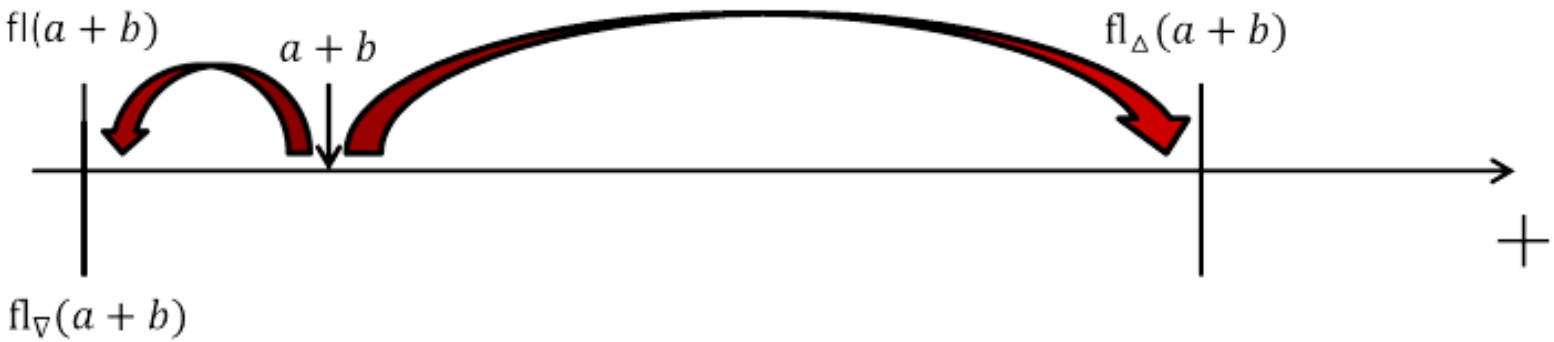
本日の講演

1. はじめに
2. 準備
3. 方向付き丸めの変更を行った行列の区間演算
4. 方向付き丸めを用いない計算
5. 連立一次方程式の精度保証付き数値計算法
6. まとめ

包含の基本アイディア

$a, b \in \mathbb{F}$,

$$\mathrm{fl}_{\bigtriangledown}(a+b) \leq a+b \leq \mathrm{fl}_{\bigtriangleup}(a+b)$$



内積・行列積への拡張

- 内積：$x, y \in \mathbb{F}^n$,

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

- 行列積：$A \in \mathbb{F}^{m\times p}$, $B \in \mathbb{F}^{p\times n}$

$$\mathrm{fl}_{\bigtriangledown}\left(AB\right) \leq AB \leq \mathrm{fl}_{\bigtriangleup}\left(AB\right)$$

- 内積，行列積が真の結果を上端・下端に包含している．
- BLAS の関数を用いることができるため，パフォーマンスを得やすい．
- ただし，ストラッセンなどの行列積の演算回数を減らすアルゴリズムは用いていないと仮定する．

行列積の区間演算の実装：丸めの変更

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 [1]：

```
system_dependent('setround', mode)
feature('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）

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．

[1] 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．

行列積の区間演算の実装：MATLAB

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

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

$$
\begin{aligned}
&\text{function } [C_{down}, C_{up}] = \mathrm{mul}(A, B)\\
&\qquad \mathrm{system\_dependent}('\mathrm{setround}', \mathrm{inf});\\
&\qquad C_{up} = A \cdot B;\\
&\qquad \mathrm{system\_dependent}('\mathrm{setround}', -\mathrm{inf});\\
&\qquad C_{down} = A \cdot B;\\
&\text{end}
\end{aligned}
$$

区間行列積の区間演算

- 高速な区間行列積の実現のために，中心・半径型の区間で計算を行う.
- 中心・半径型のため，区間が拡大されるが実用的である.

まず，上端・下端型の区間行列が与えられた場合,

- 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.

$$
\begin{aligned}
A_m &= \mathrm{fl}_{\triangle}\left((\overline{A} - \underline{A})/2 + \underline{A}\right), \\
A_r &= \mathrm{fl}_{\triangle}\left(A_m - \underline{A}\right).
\end{aligned}
$$

区間行列積の区間演算

定理

$\mathbf{A} \in \mathbb{IR}^{m\times n}$, $\mathbf{B} \in \mathbb{IR}^{n\times p}$ 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．

区間行列積の区間演算

- なぜ中心半径型?

$\mathbf{A},\ \mathbf{B} \in \mathbb{IR}^{2\times 2}$,

$$\mathbf{A} = \begin{pmatrix} [1,2] & [-1,1] \\ [0,1] & [-1,2] \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} [0,1] & [-1,1] \\ [1,2] & [-2,2] \end{pmatrix}$$

- 区間行列同士の行列積 $\mathbf{AB}$ を考える.

区間行列積の区間演算

$$
\begin{aligned}
\mathbf{AB} &= \begin{pmatrix} [1,2] & [-1,1] \\ [0,1] & [-1,2] \end{pmatrix} \cdot \begin{pmatrix} [0,1] & [-1,1] \\ [1,2] & [-2,2] \end{pmatrix} \\
&= \begin{pmatrix} [0,2]+[-2,2] & [-2,2]+[-4,4] \\ [0,1]+[-2,4] & [-1,1]+[-4,4] \end{pmatrix} \\
&= \begin{pmatrix} [-2,4] & [-6,6] \\ [-2,5] & [-5,5] \end{pmatrix}
\end{aligned}
$$

- 上端・下端型の区間行列同士では逐次的に計算していかないといけない.
- 行列同士の掛け算のように簡単に上向きの計算，下向きの計算のようにいかない.
- そのため，中心・半径型に変形し BLAS を利用できる形の定理に持ち込む.

区間行列積の区間演算の実装

上端・下端型の区間行列を中心・半径型区間行列に変換するアルゴリズム.

アルゴリズム

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

$$
\begin{aligned}
&\text{function } [A_m, A_r] = \text{trans\_midrad}(\underline{A}, \overline{A})\\
&\qquad \text{system\_dependent}('\text{setround}', \text{inf});\\
&\qquad A_m = (\overline{A} - \underline{A})/2 + \underline{A};\\
&\qquad A_r = A_m - \underline{A};\\
&\text{end}
\end{aligned}
$$

区間行列積の区間演算の実装

中心・半径型区間行列積を計算するアルゴリズム.

アルゴリズム

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

$$
\begin{aligned}
&\text{function } [\underline{C}, \overline{C}] = \text{interval\_mul}(A_m, A_r, B_m, B_r) \\
&\qquad \text{system\_dependent}('\text{setround}', \text{inf}); \\
&\qquad T = (|Am| \cdot B_m + A_r \cdot (|Bm| + Br)); \\
&\qquad \overline{C} = A_m \cdot B_m + T; \\
&\qquad \text{system\_dependent}('\text{setround}', -\text{inf}); \\
&\qquad \underline{C} = A_m \cdot B_m - T; \\
&\text{end}
\end{aligned}
$$

本日の講演

1. はじめに
2. 準備
3. 方向付き丸めの変更を行った行列の区間演算
4. 方向付き丸めを用いない計算
5. 連立一次方程式の精度保証付き数値計算法
6. まとめ

方向付き丸めを用いない計算

- 方向付き丸めの変更が困難な環境が存在する.
  - 例えば，Graphics Processing Unit (GPU).
  - 方向付き丸めを一回の四則演算毎に変更しなければいけない.
- その場合，方向付き丸めを用いない計算が有効.
  - 方向付き丸めを用いない計算 $\Rightarrow$事前誤差評価
  - 事前誤差評価による誤差の拡大が起こる.
- 高精度計算 を併用することで過大評価を抑制

方向付き丸めを用いない計算：準備

以下に本節で取り扱う記号の定義を行う．

定義

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

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

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．

方向付き丸めを用いない計算：準備

定義

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．

アルゴリズム

浮動小数点数における ufp の値を求めるアルゴリズム

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

- S.M. Rump. *Error estimation of floating-point summation and dot product,* BIT Numerical Mathematics, 52(1): 201–220, 2012.

方向付き丸めを用いない計算：準備

定義

$r \in \mathbb{F}$ 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 を得るための手法が提案されている．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．

- S.M. Rump, P. Zimmermann, S. Boldo, and G. Melquiond. *Computing predecessor and successor in rounding to nearest,* BIT Numerical Mathematics, 49(2): 419–431, 2009.

方向付き丸めを用いない計算

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．
$a, b \in \mathbb{F}$ において，加減算は

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

と表される．よって，

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

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

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

と表される．よって，

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

と表される．

総和と内積の誤差評価：クラシカル

定理

総和の誤差解析：$x \in \mathbb{F}^n$ 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

内積の誤差解析：$x, y \in \mathbb{F}^n$ において，

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

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

- Nicholas J. Higham. Accuracy and Stability of Numerical Algorithms, Second Edition, SIAM Publications, Philadelphia, 2002.

総和の誤差解析について

$s = x_1 + x_2 + \cdots + x_n$ とし，また，$s_i = x_1 + x_2 + \cdots + x_i$ とし，$\tilde{s}_i$ は浮動小数点演算によって得られた近似の値を表す（ただし，$s_1 = \tilde{s}_1 = x_1$）.

$$\begin{aligned}
\tilde{s}_2 &= \mathrm{fl}(x_1 + x_2) = (x_1 + x_2)(1 + \epsilon_1), \\
\tilde{s}_3 &= \mathrm{fl}(\tilde{s}_2 + x_3) = ((x_1 + x_2)(1 + \epsilon_1) + x_3)(1 + \epsilon_2) \\
&= x_1(1 + \epsilon_1)(1 + \epsilon_2) + x_2(1 + \epsilon_1)(1 + \epsilon_2) + x_3(1 + \epsilon_2)
\end{aligned}$$

のように誤差を評価していくことができる.

$$|s_3 - \tilde{s}_3| = |(x_1 + x_2 + x_3) - (x_1(1 + \theta_2) + x_2(1 + \theta_2) + x_3(1 + \epsilon_2))| \leq \gamma_2 \sum_{i=1}^{3} |x_i|$$

ここで $|\epsilon_i| \leq \mathbf{u}$, $nu < 1$ とし

$$\prod_{i=1}^{n}(1 + \epsilon_i) = 1 + \theta_n$$

が成り立つ．ただし,

$$|\theta_n| \leq \frac{n\mathbf{u}}{1 - n\mathbf{u}} \leq \gamma_n$$

総和と内積の誤差評価

- 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きた.
- Claude P. Jeannerod 氏，Siegfried M. Rump 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.

定理

$x \in \mathbb{F}^n$ 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

$x, y \in \mathbb{F}^n$ において,

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

が成立する.

- 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.

ufp を用いた誤差評価

- Siegfried M. Rump 氏により ufp を用いた誤差評価が提案されている．

$a, b \in \mathbb{F}$ において，加算は

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

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

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

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．

ufp を用いた総和の誤差解析

定理

$x \in \mathbb{F}^n$ 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.

ufp を用いた内積の誤差解析

定理

$x, y \in \mathbb{F}^n$ 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，

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

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

誤差評価の比較：総和の場合

定理

$x \in \mathbb{F}^n$ 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

$x \in \mathbb{F}^n$ において,

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

が成立する

- $\mathrm{ufp}(\sum_{i=1}^{n} |x_i|) \leq \sum_{i=1}^{n} |x_i|$.
- 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事が多い.
- 上限の評価が浮動小数点演算 $\mathrm{fl}(\cdot)$ の形で書かれているため計算しやすい.

方向付き丸めを用いない行列積の包含

- 行列 $A \in \mathbb{F}^{m\times n}$, $B \in \mathbb{F}^{n\times p}$ の行列積の包含を考える.
- 方向付き丸めを用いず, 中心・半径型の区間 $\langle C, R\rangle \in \mathbb{IF}^{m\times n}$ で包含を行う,

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

と表される. ただし, $2(n+2)\mathbf{u} < 1$, $e = (1,\ldots,1)^T$.

- 必要な行列積の計算は, $\mathrm{fl}\,(AB)$, $\mathrm{fl}\,(|A||B|)$ の 2 回である.
- 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, 真の結果を包含している.
- ただし, 起こりうる丸め誤差の最大を評価しているため方向付き丸めを用いた場合より過大評価になる.

方向付き丸めを用いない区間行列積の包含

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．ufp を用いた内積の定理より $2(n+2)\mathbf{u} < 1$ を仮定して中心 A_mB_m と誤差半径 T の計算を考える．まず，中心 A_mB_m は，

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

のように計算される．次に，$|A_m|B_r$ の上限を考える．

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

$(|B_m| + B_r)$ は

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

となる．

方向付き丸めを用いない区間行列積の包含

次に，$A_r T_3$ は定理 8 より

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

となる．よって定理 3 における半径 T の上限は

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

と表される．よって，

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

が得られる．

方向付き丸めを用いない区間行列積の包含

アルゴリズム

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 $[C, R] = \mathrm{interval\_mul2}(A_m, A_r, B_m, B_r)$

if $2(\mathrm{n}+2)\mathbf{u} < 1, \mathrm{error}('\mathrm{failed}'),$ end

$C = A_m \cdot B_m;$

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

$T_1 = |A_m| \cdot B_r$

$T_2 = ((n+2)\mathbf{u} \cdot \mathrm{ufp}(T_1) + \mathbf{realmin} \cdot e^T e);$

$T_3 = |B_m| + B_r;$

$T_4 = A_r \cdot T_3;$

$T_5 = ((n+2)\mathbf{u} \cdot \mathrm{ufp}(T_4) + \mathbf{realmin} \cdot e^T e);$

$R2 = (T_1 + T_2 + T_4 + T_5 + R);$

$T_6 = \mathrm{succ}(R2 + 4\mathbf{u} \cdot \mathrm{ufp}(R2));$

end

方向付き丸めを用いない区間行列積の包含

- このとき，行列積を行う箇所は,

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

- 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．
- 全体の行列積の回数は 4 回である．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．
- そのため，方向付き丸めを用いない高精度な計算を簡単に紹介する．

高精度計算

- 無誤差変換
  - $a+b=x+y$
    $$[x,\ y]=\mathbf{TwoSum}(a,\ b),\quad x=\mathrm{fl}(a+b),\ y:\text{誤差}.$$
  - $a\cdot b=x+y$
    $$[x,\ y]=\mathbf{TwoProduct}(a,\ b),\quad x=\mathrm{fl}(a\cdot b),\ y:\text{誤差}.$$
- 高精度な内積計算アルゴリズム (Ogita, Rump and Oishi)
- 高精度な行列積アルゴリズム (Ozaki, Ogita, Oishi and Rump)

- D. E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, vol. 2, AddisonWesley, Reading, MA, 1969.
- T. J. Dekker, A floating-point technique for extending the available precision, *Numer. Math.*, vol. 18, no. 3, pp. 224–242, 1971.
- T. Ogita, S. M. Rump and S. Oishi, Accurate sum and dot product, *SIAM Journal on Scientific Computing*, vol. 26, no. 6, pp. 1955–1988, 2005.
- K. Ozaki, T. Ogita, S. Oishi and 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.

無誤差変換

アルゴリズム

$a,\ b \in \mathbb{F}$ の加算における無誤差変換

function $[x, y] = \mathrm{TwoSum}(a, b)$
  $x = (a + b);$
  $z = (x - a);$
  $y = ((a - (x - z)) + (b - z));$
end

アルゴリズム

$a,\ b \in \mathbb{F}$ の乗算における無誤差変換

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

アルゴリズム

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

高精度な内積計算アルゴリズム

- 和と積の無誤差変換を用いて高精度な計算が達成されている.
- 誤差上限付きの高精度な内積アルゴリズムを紹介する.

アルゴリズム

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

高精度計算アルゴリズム

- Dot2Err は内積を計算精度の 2 倍の精度で計算し，その誤差上限を求めるアルゴリズムであった．その他にも，
- 総和を K 倍精度で計算し，その誤差上限を求めるアルゴリズム：SumK
- 内積を K 倍精度で計算し，その誤差上限を求めるアルゴリズム：DotKErr
- Faithful な結果を得る総和のアルゴリズム：Accsum

なども考案されている．

- T. Ogita, S. M. Rump and S. Oishi, Accurate sum and dot product, *SIAM Journal on Scientific Computing*, vol. 26, no. 6, pp. 1955–1988, 2005.
- S.M. Rump, T. Ogita, and S. Oishi, Accurate floating-point summation part I: Faithful rounding, *SIAM Journal on Scientific Computing*, vol. 31, no. 1, pp. 189–224, 2008.

高精度な行列計算アルゴリズム

行列 $A \in \mathbb{F}^{m \times p}$ と行列 $B \in \mathbb{F}^{p \times n}$ における行列積 AB を考える．このとき，行列 A と行列 B を以下のように無誤差で分解する．

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

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．

高精度な行列積計算アルゴリズム

分解された行列 A, B より，行列積 AB は

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

とかける．このときのポイントは各行列積

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

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．

高精度な行列積計算アルゴリズム

そのため，行列積 AB は

$$C_1 = \mathrm{fl}\left(A^{(1)}B^{(1)}\right),\ C_2 = \mathrm{fl}\left(A^{(1)}B^{(2)}\right),\ \cdots,\ \mathrm{fl}\left(A^{(s)}B^{(t)}\right) = C_{st}$$

とおくと

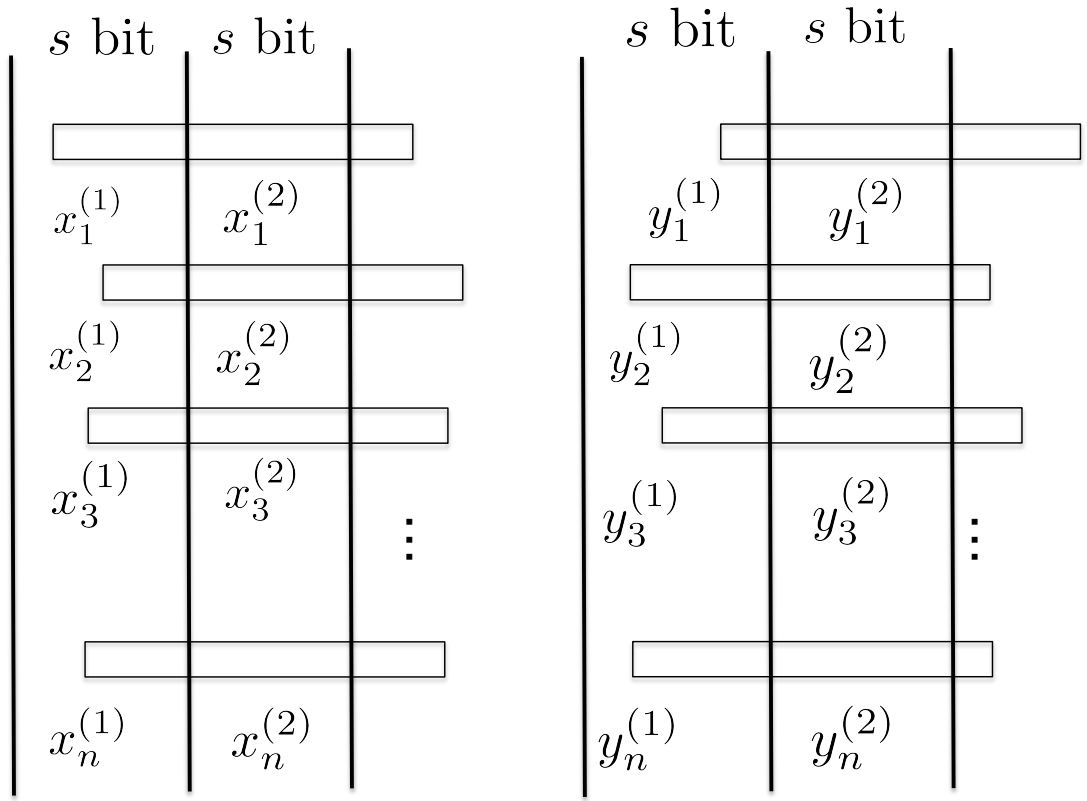
$$\begin{aligned} AB &= \mathrm{fl}\left(A^{(1)}B^{(1)}\right) + \mathrm{fl}\left(A^{(1)}B^{(2)}\right) + \mathrm{fl}\left(A^{(2)}B^{(1)}\right) + \cdots + \mathrm{fl}\left(A^{(p)}B^{(q)}\right) \\ &= C_1 + C_2 + C_3 + \cdots + C_{st} \end{aligned}$$

と変形され，行列同士の和で表される．

- 行列積の回数は st 回である．
- この和を所望の精度が得られるアルゴリズム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．
- 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．

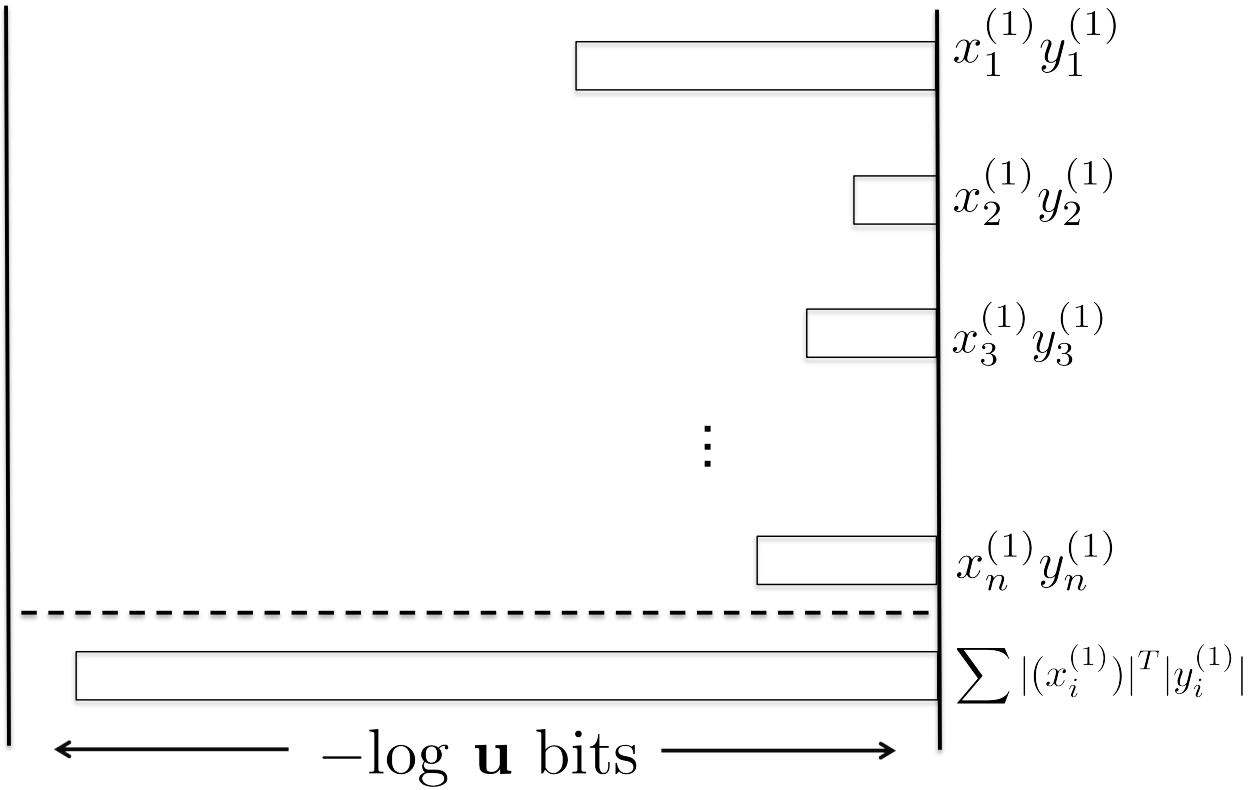
高精度な行列積計算アルゴリズム：イメージ

簡単のため，行列積内の一つの内積 $x,\ y \in \mathbb{F}$ を考える。



$$s = \left\lfloor \frac{\log_2(\mathbf{u}^{-1}) + \log_2 n}{2} \right\rfloor$$

高精度な行列積計算アルゴリズム：イメージ



分割した内積の計算結果を $-\log_2(\mathbf{u})$ bits に収まるように，分割，計算を行うことで，無誤差の計算結果を達成している。

高精度な行列積計算アルゴリズム

アルゴリズム

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

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

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

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

本日の講演

1. はじめに
2. 準備
3. 方向付き丸めの変更を行った行列の区間演算
4. 方向付き丸めを用いない計算
5. 連立一次方程式の精度保証付き数値計算法
6. まとめ

連立一次方程式の精度保証付き数値計算

- 連立一次方程式 $Ax = b,\ A \in \mathbb{F}^{n\times n},\ b \in \mathbb{F}^n$ の精度保証付き数値計算
- 係数行列に特殊な条件を仮定せず，$n = 100000$ 程度までの問題をスーパーコンピュータ上で計算する．

定理

R : A の近似逆行列
I : 単位行列

正則性の検証： $\|RA - I\|_\infty < 1,$

A^{-1} が存在し

誤差上限： $\|x - \tilde{x}\|_\infty \leq \dfrac{\|R(A\tilde{x} - b)\|_\infty}{1 - \|RA - I\|_\infty}.$

方向付き丸めを用いた計算例

Oishi and Rump (2002)

- $C := \mathrm{fl}_{\nabla}(RA - I) \leq RA - I \leq \mathrm{fl}_{\triangle}(RA - I) =: D.$
- $\mathrm{fl}_{\triangle}(\|\max(|C|, |D|)\|_{\infty}) < 1$ なら A^{-1} が存在する.
- 同様に分子式も丸め誤差を把握しながら計算を行う.

しかし，スーパーコンピュータ上でも方向付き丸めの変更が困難な場合もある.

$$\Downarrow$$

事前誤差解析と，高精度計算を組み合わせてシャープな誤差上限を得る.

$\|RA - I\|_\infty$ の上限の計算方法

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

$\max(\alpha_1) < 1$ なら,

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

where $r \in \mathbb{R},\ \mathrm{succ}(r) := \min\{f \in \mathbb{F} : r < f\},\ \mathrm{ufp}(r) := 2^{\lfloor \log_2 |r| \rfloor}$,

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

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

$\|R(A\tilde{x}-b)\|_\infty$ の上限の計算方法

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

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

ただし，$r\in\mathbb{R}$, $\mathrm{succ}(r):=\min\{f\in\mathbb{F}: r<f\}$, $\mathrm{ufp}(r):=2^{\lfloor\log_2|r|\rfloor}$,

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

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

高精度計算の導入

大規模計算を行うにあたって,

- 事前誤差評価を用いているため，過大評価されるため
  - 適応範囲が狭まる
  - 精度保証結果が悪くなる.
- 近似解の精度の悪化

$$\Downarrow$$

- $\|RA - I\|_\infty$ の上限をシャープに評価するため，Ozaki らの高精度行列積アルゴリズムを利用.
- 精度のよい近似解を得るため，残差反復 $\tilde{x}_{\mathrm{new}} = \tilde{x} - R(A\tilde{x} - b)$ における残差の計算に高精度内積計算アルゴリズム Dot2 を利用.
- $\|R(A\tilde{x} - b)\|_\infty$ の上限をシャープに評価するため，残差の計算に高精度内積計算アルゴリズム Dot2Err を利用.

実験環境

Fujitsu PRIMEHPC FX10 24nodes

- CPU:Fujitsu SPARC64$^{\mathrm{TM}}$IXfx (1.848GHz/16-core) /1node, in total 5.676TFLOPS
- Memory:32Gbytes/1node, in total 768Gbytes
- C compiler: Fujitsu C/C++ Compiler Driver Version 1.2.1 (Jan 25 2013 17:17:00)
- Library: BLACS, PBLAS and ScaLAPACK optimized by Fujitsu.
- Compiler option: -Kopenmp -SCALAPACK -SSL2BLAMP -O0

問題 1: n のサイズを変更

$n = 10000, 30000, 50000, 80000, 100000.$
A: [0, 1] 区間の乱数行列.
$b = \mathrm{fl}(A \cdot e),\ e = (1, \ldots, 1)^T \in \mathbb{F}^n.$

Table 1: $\|RA - I\|_\infty$ の上限

n	NR	NR+Acc
10000	1.6E-05	2.0E-09
30000	1.0E-04	7.3E-09
50000	7.6E-04	4.2E-08
80000	3.5E-03	1.8E-07
100000	6.6E-03	3.4E-07

Table 2: $\|\tilde{x} - x\|_\infty$ の上限

n	NR	NR+Acc
10000	2.7E-05	1.1E-16
30000	1.1E-04	1.5E-16
50000	9.9E-04	8.5E-16
80000	5.8E-03	5.8E-15
100000	8.7E-03	1.3E-14

問題 2: 悪条件問題 $n = 100000$

A: Higham のテスト関数作成法 randsvd を利用 $n = 100000$.
$b = \mathrm{fl}(A \cdot e),\ e = (1, \ldots, 1)^T \in \mathbb{F}^n$.

Table 3: $\|RA - I\|_\infty$ の上限

cond	NR	NR+Acc
10^2	4.5E-06	4.6E-09
10^4	3.0E-04	2.3E-07
10^6	2.2E-02	1.4E-05
10^8	–	7.5E-04
10^{10}	–	1.7E-02

'–'：精度保証失敗

Table 4: $\|\tilde{x} - x\|_\infty$ の上限

cond	NR	NR+Acc
10^2	4.3E-06	1.1E-16
10^4	1.9E-04	1.2E-16
10^6	1.6E-02	1.0E-15
10^8	–	8.5E-14
10^{10}	–	8.1E-12

- N. J. Higham. Accuracy and stability of numerical algorithms. SIAM Publications, Philadelphia, 2nd edition, 2002.

Table 5: 計算時間の比較 (sec)

n	INV	MM	NR	NR+Acc
10000	10.32	1.29	11.55	18.94
30000	89.83	14.34	104.70	169.18
50000	477.22	61.63	540.27	818.96
80000	806.89	218.75	1028.23	1684.36
100000	1894.43	455.75	2355.04	3805.52

INV: 近似逆行列を求める計算時間
MM: 行列積を求める計算時間

本日の講演

1. はじめに
2. 準備
3. 方向付き丸めの変更を行った行列の区間演算
4. 方向付き丸めを用いない計算
5. 連立一次方程式の精度保証付き数値計算法
6. まとめ

まとめ

本講演では

- 方向付き丸めを用いた行列積・区間行列積の区間演算
- 浮動小数点演算における事前誤差解析
- 方向付き丸めを用いない行列積・区間行列積の区間演算
- 無誤差変換を用いた高精度計算
- 連立一次方程式の精度保証付き数値計算法への応用

について紹介させて頂きました.

まとめ

ご清聴ありがとうございました．