計算機援用証明と精度保証付き数値計算

荻田 武史
東京女子大学 現代教養学部 数理科学科

東京大学 大学院数理科学研究科
応用数理特別講義II
2016年7月21日

講義の概要

数学の定理を証明するために、コンピュータによる計算を用いる方法を「計算機援用証明（computer-assisted proof）」と呼ぶ。

この講義の前半では、計算機援用証明に関する実例を紹介する。

後半は、それを支える技術である「精度保証付き数値計算」について触れる。

前半の講義内容

1. コンピュータを用いた証明の例（成功例）

2. 証明とは何か？

3. コンピュータの計算に関する失敗例

内容は下記の文献に基づく

Siegfried M. Rump （荻田 武史訳）:
[1] 計算機援用証明 I, 応用数理, 14:3 (2004), pp. 214–223.
[2] 計算機援用証明 II, 応用数理, 14:4 (2004), pp. 346–359.

コンピュータを用いた証明の例（成功例）

- メルセンヌ素数の探索
- 四色問題（四色定理）の解決
- Risch（リッシュ）のアルゴリズム
- ケプラー予想（ヒルベルトの第18問題）の解決
- など

メルセンヌ素数の探索

メルセンヌ数：$M_n = 2^n - 1$ $(n = 1, 2, \dots)$

メルセンヌ素数：素数であるメルセンヌ数

例：

大昔：　$2^2 - 1$（1番目）

1971年：　$2^{19,937} - 1$（10進数で約6,000桁。24番目）
Bryant Tuckerman, IBM Thomas J. Watson Center

2015年：　$2^{74,207,281} - 1$（10進数で約2,200万桁。49番目？）
GIMPS: Great Internet Mersenne Prime Search

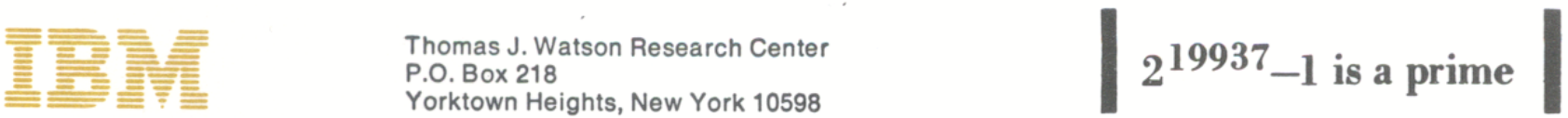


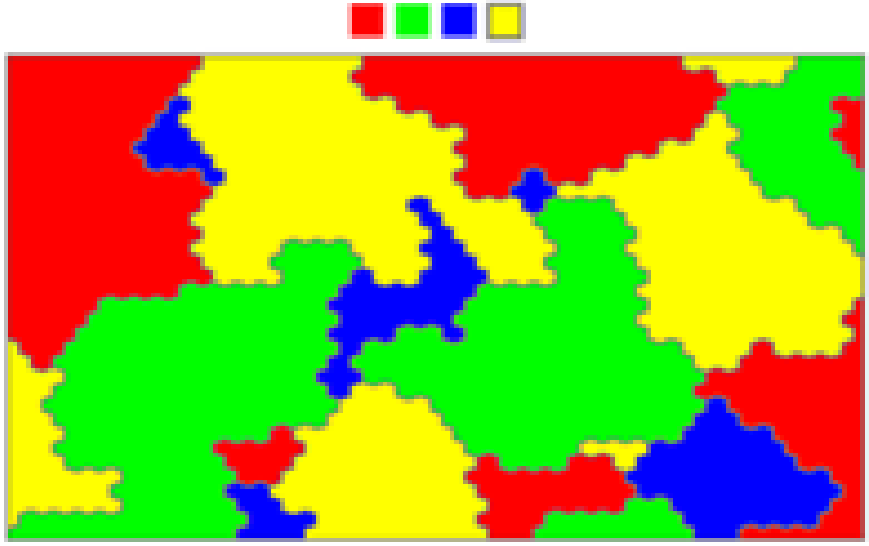
Figure 1: IBM Thomas J. Watson Centerの郵便封筒に印刷されたもの

現在GIMPSでは、Lucas-Lehmerテストというコンピュータ向きの判定法を用いて、世界中で（インターネットを介して数十万台のパソコンを用いながら）メルセンヌ素数の探索が続けられている。（みなさんも自由に参加できます。賞金も出ます：現在3,000 USドル、10進数で1億桁以上のものを見つけると、50,000 USドル）

四色問題（四色定理）の解決

「（ある条件の下で）いかなる地図も、隣接する領域が異なる色になるように塗るには4色あれば十分である」

19世紀に問題が定式化されて以来、100年以上未解決だった。



四色定理は、1976年にKenneth AppelとWolfgang Hakenによってコンピュータを用いて「証明」された。（この証明方法には、莫大な計算処理が必要だった）

Figure 2: イリノイ大学アーバナ・シャンペーン校の郵便封筒に印刷されたもの

Risch（リッシュ）のアルゴリズム

「与えられた（初等関数の組み合わせから成る）関数の積分が、初等関数の組み合わせで表現可能かどうか」を判定するアルゴリズム。もし可能なら、その積分の式を計算する。

例：

$$\int e^{x^2} dx \quad \text{は初等関数で表現できない}$$

$$\Longleftarrow \quad \int e^{x^2} dx = -\frac{1}{2}\sqrt{-\pi} \cdot \mathrm{erf}(\sqrt{-1} \cdot x) + C$$

$$\text{（誤差関数）} \quad \mathrm{erf}(x) := \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du$$

ケプラー予想（ヒルベルトの第18問題）の解決

1661年（Johannes Kepler）：「立方体の内部に同じ大きさのボールを充填するとき、面心立方格子（立方最密構造）の充填が最も高密度な充填になる」

$$\text{充填率：} \frac{\pi}{3\sqrt{2}} \approx 74\% \text{（六方最密構造も同じ充填率）}$$

この300年以上昔の予想は、1998年にThomas C. Halesによって（肯定的に）解決された。その証明ではいくつかの大きな非線形最適化問題を厳密に解く必要があった。

$\Longrightarrow$ 証明の一部に、精度保証付き数値計算が用いられた。

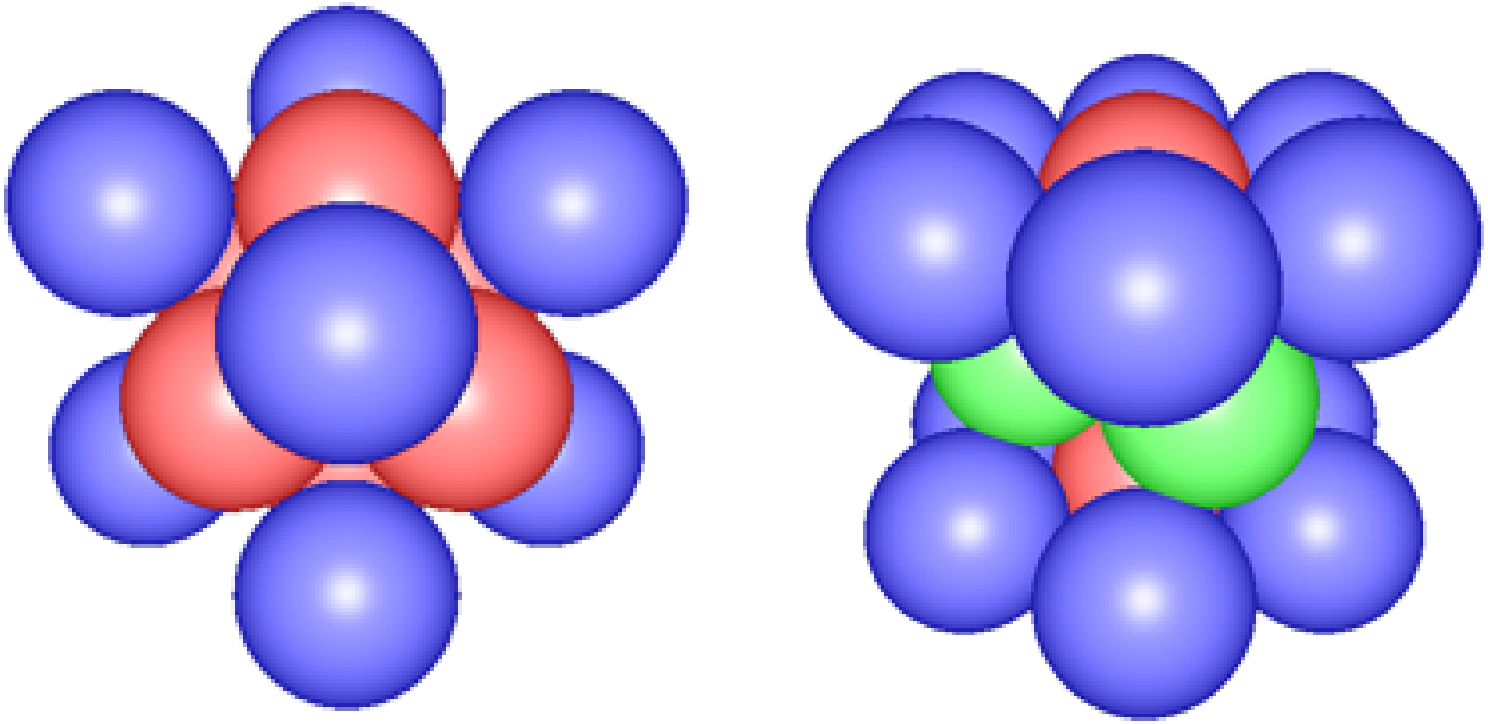


Figure 3: 立方最密構造（左）と六方最密構造（右）

画像作成・提供：尾崎克久様（芝浦工業大学）

証明とは何か？

これまで示した例は、それぞれまったく異なる種類のものであるが、共通してコンピュータによる莫大な計算過程を必要とする。

コンピュータを用いた証明は信頼できるのか？

$\Downarrow$

どういった場合に、そのような証明は信頼できるのか？

$\Downarrow$

証明とは何か？

証明とは何か？

- 証明が人間により行われ、人間によって確認されたということが正しさを保証する？
  $\Longrightarrow$　数学者も人間なので、間違った結論を出すこともある

- 証明にコンピュータを使用したことが、証明の正しさを損ねるのだろうか？

難しい証明は『受け入れられる』ために、しばしば（十分な人数の信頼できる数学者によって確かめられるために）かなりの時間を必要とする。

Frank N. Cole氏による『講演』

1903年10月ニューヨークでのAMS（アメリカ数学会）の会議で、Coleはとても変わった『講演』をした。

$$761838257287 \times 193707721 \quad =$$

Frank N. Cole氏による『講演』

1903年10月ニューヨークでのAMS（アメリカ数学会）の会議で、Coleはとても変わった『講演』をした。

$$761838257287 \times 193707721 \quad = \quad 147573952589676412927$$

Frank N. Cole氏による『講演』

1903年10月ニューヨークでのAMS（アメリカ数学会）の会議で、Coleはとても変わった『講演』をした。

$$2^{67} - 1 \quad =$$

Frank N. Cole氏による『講演』

1903年10月ニューヨークでのAMS（アメリカ数学会）の会議で、Coleはとても変わった『講演』をした。

$$2^{67} - 1 \quad = \quad 147573952589676412927$$

Frank N. Cole氏による『講演』

1903年10月ニューヨークでのAMS（アメリカ数学会）の会議で、Coleはとても変わった『講演』をした。

$$\begin{aligned} 761838257287 \times 193707721 &= 147573952589676412927 \\ 2^{67} - 1 &= 147573952589676412927 \end{aligned}$$

$\Longrightarrow$　このメルセンヌ数は合成数！

（合成数であることは既に知られていたが、具体的な因数は見つかっていなかった）

（ちなみに、電卓の登場は、1960年代前半。これを見つけるのに「毎週日曜で3年」掛かったらしい）

現代では ‥

今日、何人の人が筆算で先程のような掛け算をするだろうか？

⇓

多くの人が少なくとも電卓等を使うか、あるいはMathematicaやMapleなどの計算機代数システムを使用するのではないだろうか？

⇓

証明の一部に、この程度の電子機器を使用することに抵抗を感じる人は、現在ではほとんどいないと言ってよいだろう。

電子計算機に対する信頼のレベル

数学の証明を支援する目的で

- 電卓等を使用することは、大抵受け入れられる
- 計算機代数システム（Mathematica、Maple等）を使用することは、それなりに受け入れられる
- コンピュータの浮動小数点演算（有限桁計算）を使用することは、かなり疑わしい

と思うのは、きっと常識的なことであろう。

思い込みによる「常識」

例えば、サメとミツバチを比べたら、サメのほうが怖いイメージがあるだろう。

サメに殺される確率のほうが高い？

$$\Updownarrow$$

統計データから、実際にはサメに殺されるよりもミツバチに殺される事例のほうが断然多いことがわかる。

思い込みによる「常識」

電卓とコンピュータでは、どちらでエラーが起こりやすい？

⇓

エラーの可能性は、システムの複雑さに比例するのだろうか？

⇓

電卓の計算では、大きなエラーは起こらない？

現在のほとんどの単純な電卓で起きる例

標準的な8桁の電卓で、以下の計算をする。

$$1\,000\,000 - 999\,999.95$$

電卓による答えは、0.1となる。

（一般的に、N桁の電卓のときは、0は$N-2$個並べれば良い）

（スマートフォン等のソフト的な電卓では、大抵正しく計算されます）

現在のほとんどの単純な電卓で起きる例

標準的な8桁の電卓で、以下の計算をする。

$$1\,000\,000 - 999\,999.95$$

電卓による答えは、0.1となる。

1 000 000	
999 999.9	5
0.1	

コンピュータの計算は？

電卓のような単純な機械でさえ、まったく誤った結果を出すことがあるなら、一体、パソコンやスーパコンピュータによって信頼できる結果を得ることができるのだろうか？

⇓

浮動小数点演算のような有限桁の計算で得られる結果は『本質的に』信頼できないのでは？

昔のコンピュータによる典型的な計算間違い

たとえば、1980 年代まで、UNIVAC 1108 のような大型コンピュータは、浮動小数点演算で

$$16777216 - 16777215 = 2$$

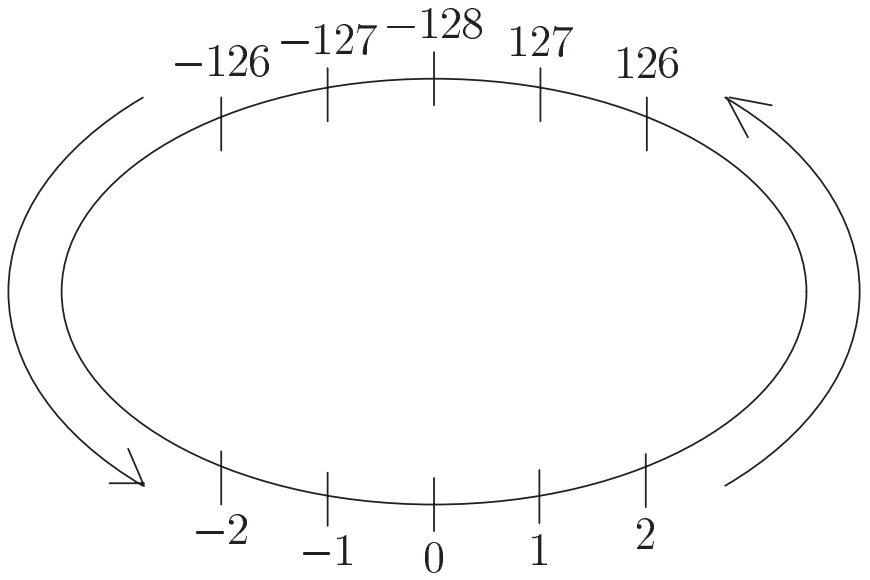
と計算した。

同様の計算間違いが、1960 年代前半に IBM の S/360 という大型コンピュータでも起こっていたが、同社はこれを改善するためにアーキテクチャを変更した（主に William M. Kahan の尽力による）。

整数演算におけるラップアラウンド（wrap-around）

8ビットの符号付整数（$-128 \le x \le 127$）において、$k = 0$から始めて、kを1ずつ増加させていくとどうなるか（$k \leftarrow k + 1$）。

$$
\begin{aligned}
1 & \\
2 & \\
3 & \\
\vdots & \\
126 & = 2^7 - 2 \\
127 & = 2^7 - 1 \\
-128 & = -2^7 \\
-127 & = -2^7 + 1 \\
-126 & = -2^7 + 2
\end{aligned}
$$



(demo)

コンピュータの計算に関する失敗例

- 湾岸戦争での出来事（1991年）
- アリアン5ロケットの事故（1996年）
- など

湾岸戦争での出来事（1991年）

湾岸戦争中、米軍の兵舎を狙って発射されたスカッドミサイルを、パトリオットミサイルによって迎撃しようとしたところ失敗し、結果として28名の兵士の命が失われた。システム起動からの経過時間を計測する部分に演算誤差が混じっていたのが、迎撃に失敗した原因とされている。

$\Longrightarrow$ 10進数の0.1を2進数に変換したのが原因

$\Longrightarrow$ 10進数の0.1は、2進数では$0.0001100110011\cdots$となり、有限桁で表現できない

アリアン5ロケットの事故（1996年）

アリアン5ロケットは，欧州宇宙機関（ESA）等が10年近くの歳月と約70億ドルの費用を投入して開発したロケットだった。しかし、このロケットは離陸から約40秒後に、約5億ドル相当もの機器を積載したまま爆発し、打ち上げは失敗に終わった。

事故の原因は、慣性基準装置内にあるソフトウェアが、水平方向の速度値を64ビットの倍精度浮動小数点数から16ビットの符号付整数へ変換していたことにあるとされている。

$\Longrightarrow$　16ビット整数値の範囲は、$-32768 \leq x \leq +32767$

$\Longrightarrow$　この変換は、簡単にエラーを引き起こす

責められるべきなのは、コンピュータの演算？

ここまでで得られる教訓

- 人は作業をしている環境を正確に把握していなければならない
- あらゆるツールはその限界をもっており、ユーザはそれらを正確に知らなければならない
- ツールは正しい方法で使わないと、予想外かつ間違った結論が生じることがある

$\Longleftrightarrow$　しかし、ユーザが誤解している危険性もある（たとえば、「整数の加減算は常に正確」という思いこみ）。

どうすれば安全なソフトウェアを設計できるのか？

1. コンピュータの計算に信頼できるルールを作る（規格の標準化）
2. ソフトウェアの作成者は、その規格を熟知する
3. 信頼できるソフトウェアを作成する

後半の講義内容

1. IEEE 754 浮動小数点演算規格

2. 精度保証付き数値計算の概要

計算機シミュレーション

流体運動などの物理現象におけるシミュレーション手順

(i) 現象を数理モデル化（現象 $\Rightarrow$ 微分方程式，モデル化誤差）

(ii) 微分方程式を離散化（微分方程式 $\Rightarrow$ 代数方程式，離散化誤差／丸め誤差）

(iii) 代数方程式を解く（代数方程式 $\Rightarrow$ 近似解，打ち切り誤差／丸め誤差）

精度保証付き数値計算の直接的な適用対象は(ii),(iii)

IEEE 754浮動小数点演算規格

浮動小数点演算： 簡単に言えば、有限桁（たとえば、10進数で16桁程度）の計算のこと。何かの計算をする度に、決められた桁数以下の部分は丸められる。

IEEE 754-1985： カリフォルニア大学バークレー校のKahanを中心として1985年に制定された2進浮動小数点演算の規格。

単精度（32ビット）や倍精度（64ビット）のフォーマット・演算のルール・例外処理等が定義されている。

たとえば倍精度であれば，符号（1ビット），指数部（11ビット），仮数部（52ビット）。浮動小数点数は正規化（仮数部を1.xxxxxxの形にする）を前提に考えるので、仮数部は53ビット分確保。

倍精度の演算精度は、$2^{-53} \approx 1.11 \times 10^{-16}$ となるため、10進数でおよそ16桁程度。

現在は、改訂版のIEEE 754-2008がある。

IEEE 754では、浮動小数点数の四則演算をsemimorphismと呼ばれる性質によって定義している。

まず実数から浮動小数点数への丸め（写像）を定義する。

IEEE 754では丸めモードとして、以下の4種類がある：

- 最近点への丸め
- $+\infty$ 方向への丸め
- $-\infty$ 方向への丸め
- 原点方向への丸め

$\mathbb{R}$： 実数の集合
$\mathbb{F}$： 浮動小数点数の集合

$r \in \mathbb{R}$に対し，それぞれの丸めモードは以下のような$\mathbb{R}$から$\mathbb{F}$への写像として定義される：

最近点への丸め $\square : \mathbb{R} \to \mathbb{F}$は，$\square(r)$を$r$に最も近い$f \in \mathbb{F}$とする．

$+\infty$**方向への丸め** $\triangle : \mathbb{R} \to \mathbb{F}$は，$\triangle(r)$を$f \geqq r$を満たす$f \in \mathbb{F}$の中で$r$に最も近いものとする．

$-\infty$**方向への丸め** $\triangledown : \mathbb{R} \to \mathbb{F}$は，$\triangledown(r)$を$f \leqq r$を満たす$f \in \mathbb{F}$の中で$r$に最も近いものとする．

原点方向への丸め $\diamond : \mathbb{R} \to \mathbb{F}$は，$\diamond(r)$を、$f \cdot r \geq 0$かつ$|f| \leqq |r|$を満たす$f \in \mathbb{F}$の中で$r$に最も近いものとする．

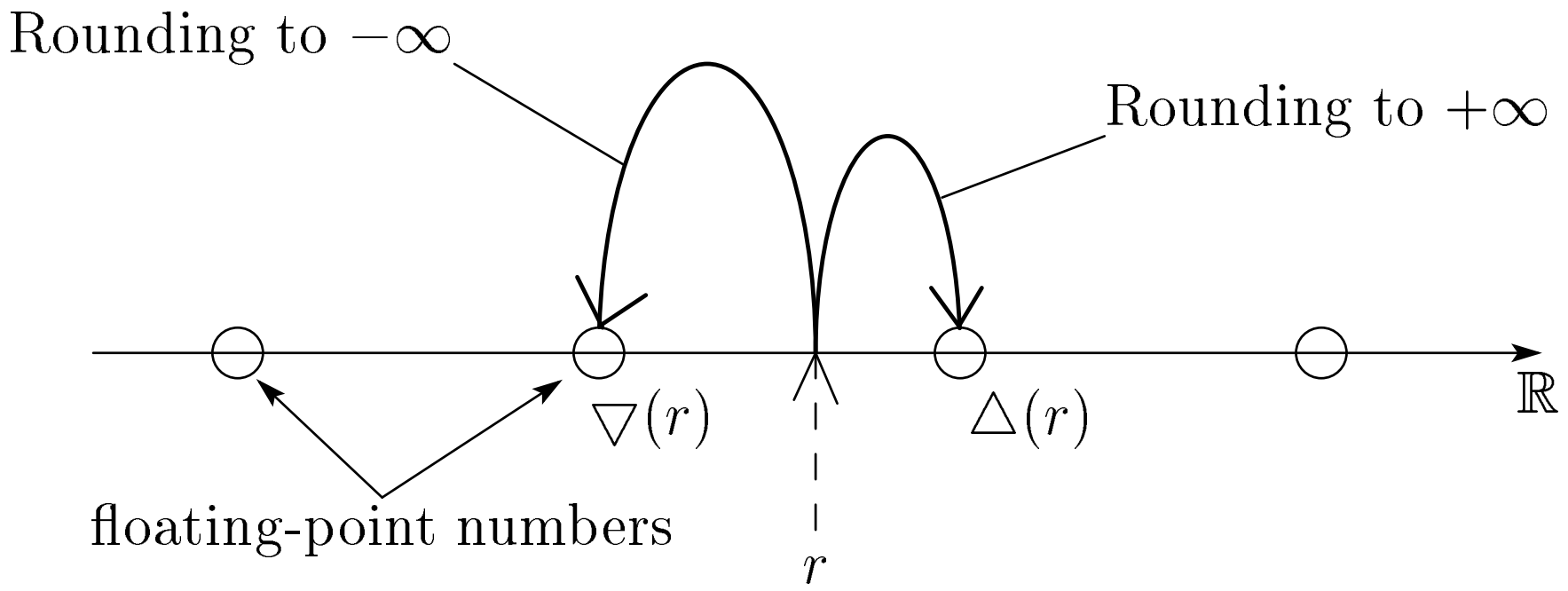


Figure 4: 丸めモードのイメージ．数直線上の ◯ は離散的に分布している浮動小数点数を表している．

次に、各丸めモードごとに浮動小数点数の四則演算を定義する。

丸めモードを指定したとき、その下での浮動小数点数の四則演算は以下のように定義される。たとえば、$a, b \in \mathbb{F}$ に対して、丸めモード $\bigcirc \in \{\square, \triangle, \triangledown, \diamond\}$ での加算 $\oplus$ は実数上の加算 $+$ を使って

$$a \oplus b = \bigcirc(a + b)$$

が満たされるように定義される。

減算や乗除算についても同様。

IEEE 754規格があると何がうれしいか？

誤差解析の観点から言えば、近似式ではなく不等式が使えるようになる。

任意の$a, b \in \mathbb{F}$に対して、最近点への丸めだけでは

$$\square(a+b) \approx a+b$$

丸めモードをうまく利用すると

$$\triangledown(a+b) \leqq a+b \leqq \triangle(a+b)$$

$\Longrightarrow$　丸め誤差に対して、厳密な議論が可能となる。

IEEE 754規格があると何がうれしいか？

- 現在、ほとんどのコンピュータは、IEEE 754規格に準拠している。
- コンピュータによる計算のルールを数学的（理論的かつ厳密的）に捉えられるようになる。
- コンピュータによる計算によって発生する誤差を定量的に計ることができるようになる。

$\Longrightarrow$　信頼性の高いソフトウェアを作成可能となる。

数値計算

数値計算： 解析的に解くのが困難な問題を数値的に解く計算やその手法。その計算にはコンピュータによる浮動小数点演算を用いるのが普通。

$\Longrightarrow$ 計算を重ねると次第に計算誤差が蓄積していく

$\Longrightarrow$ 最終的に得られた結果がどれくらい正しいかは問題依存

$\Longrightarrow$ 正しい結果とはまったく違った計算結果が得られることもしばしばある

丸め誤差の影響

前述のように，浮動小数点演算は有限桁の計算であるため丸め誤差が発生するが，実際にどのような影響があるか，いくつかの例を挙げる。

- Rumpの例題
- 2元連立一次方程式

Rumpの例題

一般に，数値計算では演算精度が高いほど結果の精度も高くなる傾向がある（例外もある）。

⇓

ある演算精度で何らかの計算をして，次にそれよりも高い演算精度で同じ計算をしたときに，双方の結果が近ければ，ある程度は結果の正しさが確認できる？

1980年代に，S. M. Rumpは次のような例題を考案した[6]。

$$f(x,y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y}$$

に，$a = 77617,\ b = 33096$ を代入した $f(a,b)$ の値を評価する。これをIBMのメインフレームS/370上で演算精度を変えて実行すると，以下のような結果となった[1]。

$$\begin{aligned}
\text{単精度（10進約8桁）：}&\quad f(a,b) \approx 1.172603\cdots \\
\text{倍精度（10進約17桁）：}&\quad f(a,b) \approx 1.1726039400531\cdots \\
\text{拡張精度（10進約34桁）：}&\quad f(a,b) \approx 1.172603940053178\cdots
\end{aligned}$$

[1]現代のIEEE 754に従うコンピュータ上で試す場合は
$(333.75 - x^2)y^6 + x^2(11x^2y^2 - 121y^4 - 2) + 5.5y^8 + x/(2y)$
のように計算手順に修正が必要である[5]。

この結果から，それぞれの精度において，一見，途中の桁までは正しい値が得られているように思われる。

実は真の値

$$f(a,b) = -0.827386\cdots$$

であり，符号も合っていない間違った結果となっている。

このように，経験則では対処できない問題もある。

2元連立一次方程式

以下のような連立一次方程式$Ax = b$を考えてみよう [4]。

$$A = \begin{pmatrix} 64919121 & 159018721 \\ 41869520.5 & 102558961 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

このとき，Aは正則で真の解は

$$x = A^{-1}b = \begin{pmatrix} 205117922 \\ 83739041 \end{pmatrix}$$

ガウスの消去法（IEEE 754の倍精度浮動小数点演算）による近似解$\widetilde{x}$は

$$\widetilde{x} = \begin{pmatrix} 106018308.0071325 \\ -43281793.0017831 \end{pmatrix}$$

ところが，この$\widetilde{x}$に対して残差rを倍精度で計算すると，残差の近似は$\widetilde{r} = (0,0)^T$となり，一見$\widetilde{x}$は正しい解のように見えてしまう

すなわち，残差の計算からでは解が正しいかどうかを判定することができないことがわかる。

(demo)

MATLABで試す場合は、$b = (1,1)^T$にすると良い。

```
>> A=[64919121, 159018721; 41869520.5, 102558961];
>> b=[1; 1];
>> x=A\b, r=b-A*x
>> xs=sym(A)\sym(b) % Symbolic Math Toolboxが必要
```

精度保証の必要性

連立一次方程式 $Ax = b$ の数値解 $\widetilde{x}$ の精度

残差 $r := b - A\widetilde{x}$ のノルム $\|r\|$ が，ある程度小さければ，$\widetilde{x}$ は十分に精度が良いのでは？

- 残差のノルムは計算が容易.
- 反復解法の反復停止条件にもよく利用されている.

$\Longrightarrow$ それでは不十分な場合がある

（例）ガウスの消去法

Aがどれだけ悪条件であったとしても，ガウスの消去法が途中で破綻せずに数値解$\widetilde{x}$が得られた場合，以下が成立：

$$\frac{\|b - A\widetilde{x}\|}{\|A\|\|\widetilde{x}\|} \sim \mathbf{u} \tag{1}$$

- LINPACKベンチマークでは，この性質を利用してプログラムが正しく実装されているかどうかをチェックしている．
- 相対残差ノルム$\dfrac{\|b - A\widetilde{x}\|}{\|b\|}$などの評価でも，同様のことが起きる．

(demo)

W. M. Kahanの言葉

"Significant discrepancies (between the computed and the true result) are very rare, too rare to worry about all the time, yet not rare enough to ignore."

直訳 浮動小数点演算によって得られた結果と真の結果に大きな乖離が生じることは非常に稀であり、常に心配するにはあまりにも稀であるが、だからと言って無視できるほど稀なわけではない。

数値計算による結果を、数学的に厳密な誤差限界と共に与えることを精度保証付き数値計算という。

以下は，実際に精度保証付き数値計算によって計算機援用証明に成功した代表的な例である。

- ローレンツ・アトラクターの存在検証 [9]
  （Smaleの第14番目の問題）
- ケプラー予想（球充填問題）の肯定的解決 [1]
  （約400年間の未解決問題）
- Double Bubble予想の肯定的解決 [2]
  （100年間以上の未解決問題）

最古（？）の区間演算

アルキメデスが，円に接する正多角形で挟み込んで円周率πを計算したことは有名である。

直径1の円に外接する正N角形の周の長さをP_N，内接する正N角形の周の長さをQ_Nとする。

- $Q_N \leq \pi \leq P_N$
- 同じ円に外接する正$2N$角形の周の長さは$P_{2N} = \dfrac{2P_N Q_N}{P_N + Q_N}$
- 内接する正$2N$角形の周の長さは$Q_{2N} = \sqrt{P_{2N} Q_N}$

この方法を採用して，さらに区間演算を用いてπ**の範囲を特定する**ことを考えてみよう。

10**進**6**桁の有効桁数**で計算することにして、試してみよう。

直径1の円に外接する正六角形の周の長さは$P_6 = 2\sqrt{3}$,
内接する正六角形の周の長さは$Q_6 = 3$

$$\sqrt{3} \in [1.73205, 1.73206] \;\Rightarrow\; P_6 \in [3.46410, 3.46412]$$

$$\Rightarrow\; \pi \in [Q_6, \overline{P_6}] = [3, 3.46412]$$

$$P_{12} = \frac{2P_6Q_6}{P_6 + Q_6} \in [3.21537, 3.21541],$$
$$Q_{12} = \sqrt{P_{12}Q_6} \in [3.10581, 3.10584]$$

$$\Rightarrow\; \pi \in [\underline{Q_{12}}, \overline{P_{12}}] = [3.10581, 3.21541]$$

同様に作業を続けていくと，$\pi \in [\underline{Q_{96}}, \overline{P_{96}}] = [3.14076, 3.14313]$

$\Rightarrow$ πの近似値として，3.14までは正確であることが区間演算によって「証明」された。

区間ガウスの消去法

通常のガウスの消去法におけるすべての演算を区間演算に置き換える.

$\Longrightarrow$ 区間ガウスの消去法（IGA）.

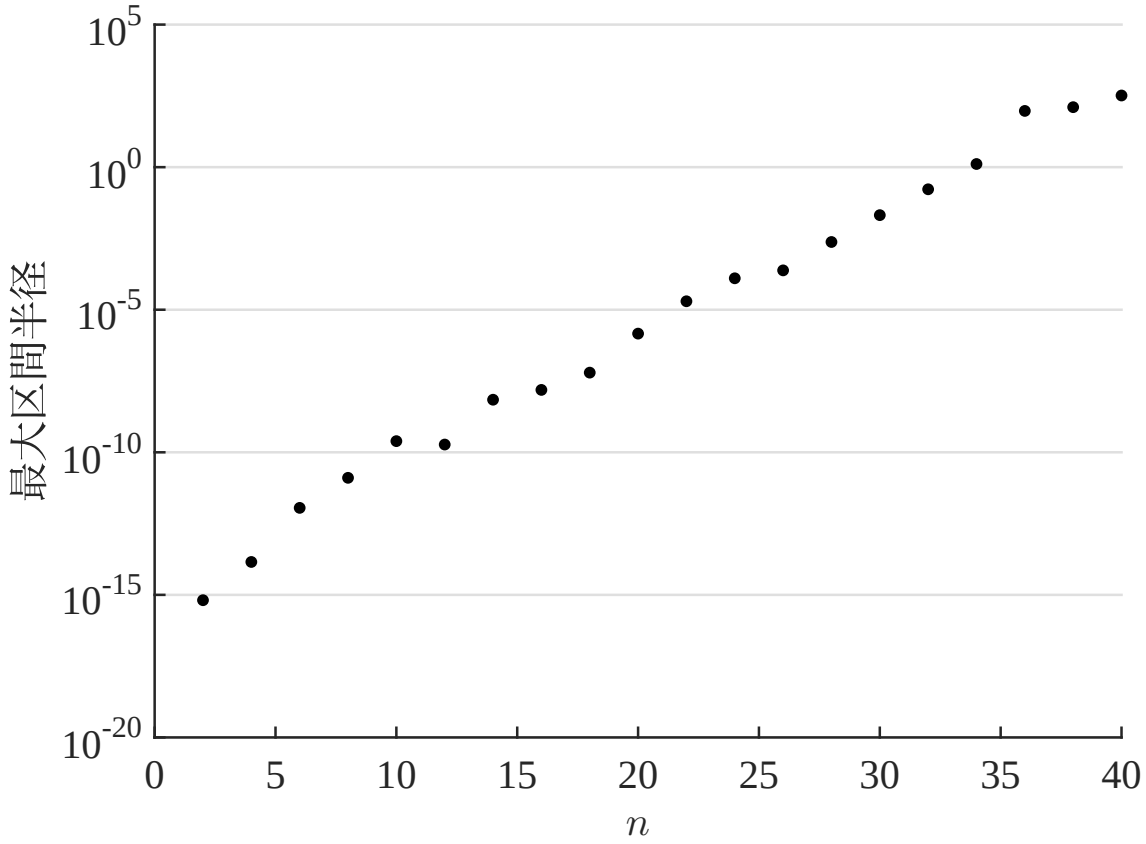
$\Longrightarrow$ 連立一次方程式の厳密解の存在範囲を得ることが（一応）可能.

$\Longrightarrow$ 特別な場合を除いて，行列の次数nがある程度以上の大きさになると，行列の条件数とは無関係に適用不可（ゼロ割の発生）.

$\Longrightarrow$ 区間演算による区間幅の増大に起因する問題.

$\Longrightarrow$ 計算速度も非常に低速.

（例）A: n次の擬似乱数行列, b: $x^* = e$となるように設定.



IGAによる解の包含$\underline{x} \le x^* \le \overline{x}$の最大半径 $\max_{1 \le i \le n} (\overline{x}_i - \underline{x}_i)/2$

J. H. Wilkinsonの言葉

区間演算の振る舞いに関連するWilkinsonの言葉[10]

> This does not imply that interval arithmetic is useless, but it does place severe restrictions on the way it can be applied. In general it is best in algebraic computations to leave the use of interval arithmetic as late as possible so that it effectively becomes an a posteriori weapon.

区間演算は，その使用をできるだけ後回しにすることができれば，有効な計算手段になり得る，ということを示唆.

$\Longrightarrow$ 高速精度保証法

最後に，今後は以下の仮定の下で議論を進めていくことを述べておく。

1. 浮動小数点演算の実装が，実際にIEEE 754規格による定義に従っている。

2. コンパイラ，オペレーティングシステムを含めて，関連のあるハードウェアやソフトウェアのすべてが正しく動作する。

References

[1] T. C. Hales: A proof of the Kepler conjecture, Annals of Mathematics, 162 (2005), 1065–1185.

[2] J. Hass, M. Hutchings, R. Schlafly: The double bubble conjecture, Electron. Res. Announc. Amer. Math. Soc. 1 (1995), 98–102.

[3] W. M. Kahan: The Regrettable Failure of Automated Error Analysis, A Mini-Course prepared for the conference at MIT on Computers and Mathematics, 1989.

[4] U. W. Kulisch, W. L. Miranker: The Arithmetic of the

Digital Computer: A New Approach, SIAM Review, 28 (1986), 1–40.

[5] E. Loh, G. W. Walster: Rump's example revisited, Reliable Computing, 8 (2002), 245–248.

[6] S. M. Rump: Algorithms for Verified Inclusions: Theory and Practice, Reliability in Computing: The Role of Interval Methods in Scientific Computing (R. E. Moore ed.), Academic Press, Boston, 1988, 109–126.

[7] S. M. Rump: Computer-assisted Proofs and Self-validating Methods, Accuracy and Reliability in Scientific Computing (Chapter 10), SIAM, Philadelphia, PA, 2005.

[8] T. Sunaga: Theory of an interval algebra and its application to numerical analysis, RAAG Memoirs, 2 (1958), 29–46.

[9] W. Tucker: The Lorenz attractor exists, C. R. Acad. Sci. Paris Sér. I Math., 328 (1999), 1197–1202.

[10] J. H. Wilkinson: Modern error analysis, SIAM Review, 13 (1971), 548–568.