現象と数学
数値計算法の数理
— 精度保証付き数値計算入門 —

桂田 祐史
http://nalab.mind.meiji.ac.jp/~mk/gensyou2020/

2020 年 9 月 3 日 (木) 10:50〜

目次

1 一応自己紹介

はじめて私の講義を聴く人もいるでしょうし。

専門は「**数値解析**」。数値解析には、

(1) 数値計算による解析　(例えば現象をシミュレーションで調べる)

という意味もあるけれど、

(2) 数値計算法の数理の解析

という意味もあり、私自身は後者の方を主にやっている。つまり数値計算の「なぜ」を追求している。

(色々な講義をしているけれど、数値解析とは直接関係がないものが多い。)

現象数理学の 3 本柱

1. モデリング … 現象を表す数式 (モデル) をつくる
2. アナリシス … モデルを数学的に調べる
3. シミュレーション … モデルをコンピューターで計算して調べる

の 3 つ目のご近所、と考えている。

2 数値計算について　なぜ数値計算するの？

(現象数理学で) なぜ数値計算をするのか？

—— M 先生が学生にするお決まりの質問だった。

現象の数理モデルの多くは微分方程式で、微分方程式は式変形では解けないものが多い (というか、ほとんどが解けないと言ってもいいくらい)。

ところがコンピューターによる数値計算ならば解けるものは多く、それで色々なことが分かる。どしどし使おう。

念のためまた質問。
なぜ「解けない」,「解ける」と違いが出るの？

おおざっぱに答え: 「解く」の意味が違う。コンピューターによる数値計算の方は近似解法で、得られるのは数値解、ある意味で基準が緩い。

「その方法で解けるのはなぜ？」「結果はどこまで信用できるの？」「その方法が有効な問題の範囲は？」…そういうのを考えるのが数値計算法の数理としての数値解析である。

2 数値計算について　「数値解析の定義」 by Trefethen

L. Trefethen [**?**] (1992) 曰く

"数値解析とは，連続数学 (continuous mathematics) の
問題に対するアルゴリズムの研究である"

これはそれ以前に、しばしば

数値解析とは、数値計算法の (丸め) 誤差の研究である

と言われたものを正したもの。誤差解析だけが数値解析のテーマではない、ということ。 —— 比較的受け入れられている見方である。

アルゴリズムの良し悪し

- 効率 (計算速度、必要な記憶容量 →扱える問題の種類・規模に影響)
- 精度
- 数値的安定性

2 数値計算について　多面的に見る (理学と工学)

私が駆け出しの頃に聞いた伊理正夫先生の言葉

「**数値計算は総合技術**」　　(ネットで [?] が読める)

は深く印象に残り、折に触れて当時のメモを読み返している。見出しだけ拾うと

1. 数値計算の研究は実験科学である
2. 数値計算と工業製品の製造とは共通する点がある
3. 数値計算は数学からは独立した分野でありまたそうあるべきである
4. 数値計算は分野横断技術である
5. 数値計算の目的は外の世界にある
6. 数値計算家はハードの極端からソフトの極端までを熟知すべし
7. 数値計算自体よりその前後の段階の方が量的には大である
8. 個人の創造性と集団の創造性とは両方とも必要である
9. 幾何学の精神と数値計算
10. 計算の品質
11. 計算幾何学と数値計算
    (以下略)

技術は工学で、理学とは違うのかもしれないが、どう違うのか、どこで重なるのか、以来ずっと気になっている。

2 数値計算について　science vs. art

私は、伊理先生の言葉を思い出すと、次の言葉が連想で思い出される。

「対称な固有値問題は science であるが、非対称な固有値問題は art である。」

実は茶色の部分を取り替えて、良く出て来る言い回し (学生の頃、はじめて読んだのが上の言葉だったので印象に残っている)。

art は「芸術」というよりも「技術」、「技」という方が近いかも。

固有値問題の数値計算というと、もう数学の話題のようだが、まだ science になってないところがある (ひょっとしてずっとそうかもしれない)、ということでもある。

3 精度保証付き数値計算 入門

数値解析の研究のトレンドは変わってきた、と私は感じている。

基本的な数値計算アルゴリズムの正当性の証明のような研究は一段落した (私見)。
コンピューターの登場からまだ 80 年弱くらい。そのため数値計算法の研究は、これまで基礎的なものが多かった、と理解している。

最近重要なこと

1. より複雑な問題に対するアルゴリズムの提唱 (こうすれば解ける)
2. **精度保証付き数値計算**

精度保証付き数値計算とは、数値計算で得たものの誤差の具体的な限界を求めたり、それによって解の「存在」や「一意性」などを数学的な意味で厳密に証明したりする、数値計算法のこと。

従来の、「とにかく (近似) 計算してみたら、こうなった」、「分割を細かくした極限ではホントの解に収束する (はず, 丸め誤差がなければ)」、「同じ位細かい分割ならばこちらの方法の精度が良くてお得 (のはず)」から前進しつつある。

数値計算を全然信用していない、という数学者が多かった。 すごくくやしい。

3.1 区間演算

3 桁の数しか使えなくても、無理数 π を、$\pi \in [3.14, 3.15]$
$(3.14 \leq \pi \leq 3.15)$ と厳密に捉えることは出来る。

Cf. アルキメデス (BC 287？–BC212) は、円に内接・外接する正多角形の周長により、$3 + \frac{1}{7} < \pi < 3 + \frac{10}{71}$ のような厳密な評価を得た。

(円周率の近似値の求め方として、正多角形の周長を用いる方法は、アルキメデス以降 1000 年近く使われた。桁数が多くなっていったが、厳密な評価を得た人は多くない。)

区間で評価 (= 不等式で評価) すれば、有限桁 (≒誤差があっても) でも、厳密な議論が出来る。
$\pi \in [3.14, 3.15]$, $e \in [2.71, 2.72]$ であるから

$$\pi + e \in [3.14 + 2.71, 3.15 + 2.72] = [5.85, 5.87].$$

$a, b \in \mathbb{R}$, $a \leq b$ を用いて $[a, b] := \{x \mid a \leq x \leq b\}$ と定められる集合を区間と呼ぶ (区間演算では、区間とは閉区間のこと、$[a, a] = \{a\}$ も区間と考える)。

3.2 試してみよう (多分分かるだろう)

ターミナルにコピペ (PDF からの場合 1 行ずつ)

```
curl -O http://verifiedby.me/kv/simple/interval-simple-0.4.48.tar.gz;
tar xzf interval-simple-0.4.48.tar.gz;
ls;
curl -O http://nalab.mind.meiji.ac.jp/~mk/misc/20200903/test1.cc;
c++ test1.cc;
./a.out
```

```
% ./a.out
1/1=[1,1]
1/2=[0.5,0.5]
1/3=[0.3333333333333333,0.3333333333333334]
1/4=[0.25,0.25]
1/5=[0.1999999999999999,0.2000000000000001]
1/6=[0.1666666666666666,0.1666666666666667]
1/7=[0.1428571428571428,0.1428571428571429]
1/8=[0.125,0.125]
1/9=[0.1111111111111111,0.1111111111111112]
1/10=[0.0999999999999999,0.1000000000000001]
```

3.3 区間演算の定義

$\mathbb{R}$ の閉区間 (interval) の全体の集合を $\mathbb{IR}$ で表す。

$$\mathbb{IR} := \{[a,b] \mid a,b \in \mathbb{R},\ a \le b\}, \quad [a,b] := \{x \in \mathbb{R} \mid a \le x \le b\}.$$

$X, Y \in \mathbb{IR}$ と演算 $*$ ($*$ は、$+$, $-$, $\cdot$ (乗法), $/$ (除法) のどれかとする) に対して

$$X * Y := \{x * y \mid x \in X,\ y \in Y\}$$

と定める。ただし $*$ が $/$ のときは、$0 \notin Y$ を仮定する ($x/0$ はマズい)。

0 を含む区間による割り算以外は、$X * Y \in \mathbb{IR}$ となる。

$X = [a,b],\ Y = [c,d]$ とするとき

$$\begin{aligned}
X + Y &:= [a+c, b+d], \\
X - Y &:= [a-d, b-c], \\
X \cdot Y &:= [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}], \\
X/Y &:= [a,b] \cdot [1/d, 1/c] \quad (\text{ただし } 0 \notin Y \text{ のとき}).
\end{aligned}$$

3.4 浮動小数点数

コンピューターでは、実数を**浮動小数点数** (floating point numbers) で近似して表すのが普通である。

細かいこと (具体的なビット表現) は省略するが、例えば Xcode の C/C++ 言語処理系の double 型のデータは、IEEE 754 binary64 という規格に準拠した浮動小数点数である (10 進法に換算して 16 桁弱の精度、絶対値が概ね $10^{-308} \sim 10^{308}$ の範囲に収まるもの)。

あるシステムの浮動小数点数全体を $\mathbb{F}$ と表す。以下は 10 進 3 桁の例で説明。

- 大きい数、小さい数は指数形式を使って表す。例えば $N_A = 6.02 \times 10^{23}$.
  指数部分 e は、例えば $-100 \leq e \leq 100$ のように制限する。
- $\mathbb{F}$ は $\mathbb{R}$ の有限部分集合である。
  $x \in \mathbb{R}$ に対して、x にもっとも近い $\mathbb{F}$ の元を $\mathrm{fl}(x)$ と表す (10 進法ならば四捨五入)。$\mathrm{fl}(\pi) = 3.14$.
- $x, y \in \mathbb{F}$ であっても、$x * y \in \mathbb{F}$ とは限らない。
  (例: 10 進 3 桁の 2 数の積 $3.14 \cdot 2.71 = 8.5094$ は、3 桁で表せない。)
- $\mathbb{F}$ での演算は (丸をつけて) $\oplus, \ominus, \odot, \oslash$ で表すことにする。$\circledast$ は、次のように定める。

$$x \circledast y := \mathrm{fl}(x * y).$$

例えば $3.14 \odot 2.71 = \mathrm{fl}(3.14 \cdot 2.71) = \mathrm{fl}(8.5094) = 8.51$ となる。まあ自然。

3.5 機械区間演算

コンピューターで処理する場合、区間の端点は $\mathbb{F}$ の元を使うしかないだろう。

$$\mathbb{IF} := \{[a, b] \mid a, b \in \mathbb{F},\ a \le b\}.$$

問題: $X, Y \in \mathbb{IF}$ のとき $X * Y \in \mathbb{IF}$ とは限らない

$X = [3.14, 3.15],\ Y = [2.71, 2.72]$ のとき、$X \cdot Y = [8.5094, 8.568] \notin \mathbb{IF}$. 一方、$[\mathrm{fl}(8.5094), \mathrm{fl}(8.568)] = [8.51, 8.57] \not\supset X \cdot Y$. 漏れてしまうかも。

解決策は、区間の左端 (下限) を求めるときは切り下げた $3.14\triangledown 2.71 = 8.50$, 区間の右端 (上限) を求めるときは切り上げた $3.15\triangle 2.72 = 8.57$ を使うことである。

$$[8.5094, 8.5680] \subset [8.50, 8.57], \quad [8.50, 8.57] \in \mathbb{IF}.$$

$X * Y$ の代わりに、$X * Y \subset X \circledast Y,\ X \circledast Y \in \mathbb{IF}$ となるような $X \circledast Y$ を計算できるようにしておく (もちろん、$X \circledast Y$ の幅はなるべく小さい方が良いが、「もっとも小さい」という条件は課さないのが普通である)。これを**機械区間演算**と呼ぶ。

3.6 区間演算ライブラリィ (使ったからには紹介)

機械区間演算を実現するためのライブラリィを、**区間演算ライブラリィ**と読んでいる。

早い段階で (1970 年代)、区間演算は有効と考えられたが、以前は特別なハードウェア、ソフトウェアが必要 (結果として高価) であった。

現在の多くの CPU は、IEEE 754 規格に準拠していて、その規格には、切り下げ、切り上げの演算も含まれているので、現在のコンピューター用に区間演算のライブラリィを作成するのは難しくない。

MATLAB 用の **INTLAB** (`http://www.ti3.tu-harburg.de/intlab/`, by S. M. Rump) が有名である (有償なので今回は使えない)。

C++言語向けのクラスライブラリィもいくつかあり、先ほど用いたのは、柏木雅英氏作成のものである (`http://verifiedby.me/kv/simple/`)。
これは **kv** (`http://verifiedby.me/kv/`) というライブラリィの簡略化バージョンである。ここでは simplified kv と呼ぶことにする。

余談　Intel 8086 CPU 用の浮動小数演算コプロセッサー 8087 の仕様設計・開発と、IEEE 754 規格の策定は同時進行で進んだ (1970 年代後半〜1980 年代半ば)。両プロジェクトで、William M. Kahan が中心的な役割を果たした。(ウィキペディアの「ウィリアム・カハン」は結構詳しい。)

3.7 簡単な例題　二分法で方程式を厳密に解いてみる

微分積分 (入門段階の解析学) で、**中間値の定理**を学ぶ。良くある証明は、区間を半分半分にしていく、**区間縮小法**に持ち込むことである (「数学解析」で解説した)。

それは容易に方程式の解を求めるアルゴリズムに出来る。「**二分法**」(bisection method) と呼ばれる。

二分法を使って $\cos x - x = 0$ の実数解を求めてみよう。simplified kv を使う C++サンプル・プログラムを紹介する。

```
curl -O http://nalab.mind.meiji.ac.jp/~mk/misc/20200903/bisection.cpp
c++ bisection.cpp
./a.out
```

この後、例えば `0 1 1e-14` enter と入力する。

…これは比較的簡単に区間演算がうまく行くケース。

注:　方程式を解くと言うと、**Newton 法**はどうなるか、気になる人がいるかもしれない。Newton 法については、解の存在を保証する **Newton-Kantorovich の定理**があり、その定理の条件を区間演算で確認すれば、解を精度保証付きで求めるプログラムが書ける。

4 精度保証付き数値計算 その先

ここまでに述べたことはずっと以前から分かっていたことである。

その後、現在までの研究の進展で重要なものを 2 つ紹介する。

4.1 連立 1 次方程式の解の精度保証付き数値計算

連立 1 次方程式 $Ax = b$ の解法は数値計算で最も重要、としばしば言われてきた。

一方、実は「区間演算は結局はうまく行かない」と考える人も多かった。

例えば連立 1 次方程式を解くアルゴリズムとして有名な Gauss の消去法は、多くの問題に対してベストに近い方法であるが、そこに現れる四則演算を単純に区間演算に置き換えた**区間 Gauss の消去法**では、非常にしばしば区間幅の爆発的増大が起きてしまい、結果の精度が低下するだけでなく、計算途中の除算で除数に 0 を含む区間が現れ、計算がそれ以上進められなくなってしまう。

そうなる理由の説明は有意義だが、ここではそれは省略する。

解決策は 2000 年頃に提出された。今では、係数行列が密行列の場合、精度保証なしで Gauss の消去法を行う場合の 2 倍程度の手間で、解 $\boldsymbol{x}$ (これはベクトル) を区間ベクトルの形で求める方法が知られている。次のスライドで軽く解説する。

もっとも微分方程式の数値解法に現れる連立 1 次方程式は大規模な**疎行列**で、それに有効な方法はまだ見つかっていない。ある先生によると “Grand Challenge” であるとか。

4.1 連立 1 次方程式の解の精度保証付き数値計算

2000 年頃から、大石進一氏等の研究グループが次の定理 (証明簡単) に基づいた、連立 1 次方程式の精度保証付き数値計算の研究を推し進めた (大石他 [?])。

$A, R \in \mathbb{R}^{N\times N}$ とする。以下の行列のノルムは作用素ノルムとする。$G := I - RA$ が $\|G\| < 1$ を満たすならば、A は正則行列であり

$$\left\|A^{-1}\right\| \le \frac{\|R\|}{1-\|G\|}.$$

特に任意の $\boldsymbol{b}, \tilde{\boldsymbol{x}} \in \mathbb{R}^N$ に対して、$\boldsymbol{x}_* := A^{-1}\boldsymbol{b}$ とおくと

$$(\heartsuit) \qquad \|\boldsymbol{x}_* - \tilde{\boldsymbol{x}}\| \le \frac{\|R\|}{1-\|G\|}\,\|\boldsymbol{b} - A\tilde{\boldsymbol{x}}\|\,.$$

R として A^{-1} の近似、$\tilde{\boldsymbol{x}}$ として $A\boldsymbol{x} = \boldsymbol{b}$ の近似解を取って、$(\heartsuit)$ を適用する。

$R \fallingdotseq A^{-1}$ だから、$\|G\| < 1$ が期待できる。$\tilde{\boldsymbol{x}}$ を Gauss の消去法で求めると、ほぼ例外なく $\|\boldsymbol{b} - A\tilde{\boldsymbol{x}}\|$ が非常に小さくなる (有名な奇跡)。区間演算を用いて、$\|G\|, \|R\|, \|\boldsymbol{b} - A\tilde{\boldsymbol{x}}\|$ を計算して $(\heartsuit)$ を適用することで、A が正則であることの証明、近似解 $\tilde{\boldsymbol{x}}$ の誤差 $\|\boldsymbol{x}_* - \tilde{\boldsymbol{x}}\|$ の評価が数学的に厳密に可能となる。

(近似解の計算と精度保証が分離されている。後者のみ区間演算をすれば十分。)

4.2 偏微分方程式　無限次元の問題が解けるなんて

連立 1 次方程式は、有限次元の線形方程式であり、基本的と考えられるが、無限次元の非線形方程式の精度保証付き数値計算に対しても、成功があった。

特に偏微分方程式についても、解の精度保証付き数値計算に成功した例が早い時期に現れ (M. Nakao [?] (1988), M. Plum (1991)、その方向で大いに研究が進展した。

[?] では、半線形楕円型偏微分方程式の境界値問題に対して、**有限要素法**による解法をベースにして、問題を不動点型の方程式 $F(u) = u$ に帰着させ、その解の存在を **Schauder の不動点定理**を使って証明した。Schauder の定理の仮定の条件に対して、不等式を用いた十分条件を作り、区間演算を用いてその条件が成立することを確認する、という手順である。

知らない言葉ばかりで、とても理解出来ない、と思うかもしれないが、解の存在を保証する定理があり、その定理の仮定が不等式であり、不等式の成立を区間演算で証明する、という大枠は、上で紹介した二分法による方程式の解の精度保証付き数値計算と同様である。

5 これからの数値解析

これまで計算の仕方が分からなかったような問題を解くことと、これまで精度保証なし解いていた問題に対して精度保証の方法を発見すること、二つの方向で研究が進展していくだろう。

有限精度の区間演算と、数学的議論を組み合わせれば、無限次元空間の問題に対しても、しばしば精度保証付き数値計算が可能となる。うまい組み合わせ方を発見するのは簡単ではないが、挑戦する価値のある課題である。

(数値計算の結果から、数学的に確かなことは何も得られない、というのは間違いだった。)

6 レポート課題について

Oh-o! Meiji のレポート・システムを使います。

質問があれば、メール (`katurada` あっとまあく `meiji.ac.jp`) で尋ねて下さい (少なくとも 9 月 3 日の 11:40, 13:00, 14:00, 15:00 にチェックします)。

また 9 月 3 日 13:30–14:30 に Zoom ミーティングを開くので (参加の仕方は Oh-o! Meiji の授業資料に書きます)、そこで質問してくれても構いません。

付録: Gauss の消去法 (知っている人も多い？念のため)

$$(1) \qquad \begin{cases} 2x_1 + 3x_2 - x_3 = 5 \\ 4x_1 + 4x_2 - 3x_3 = 3 \\ -2x_1 + 3x_2 - x_3 = 1. \end{cases}$$

前進消去

$$\begin{pmatrix} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ -2 & 3 & -1 & 1 \end{pmatrix} \to \begin{pmatrix} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 6 & -2 & 6 \end{pmatrix} \to \begin{pmatrix} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 0 & -5 & -15 \end{pmatrix}$$

後退代入

最後の行列は

$$2x_1 + 3x_2 - x_3 = 5, \quad -2x_2 - x_3 = -7, \quad -5x_3 = -15$$

ということを表しているので、後の方から順に

$$x_3 = \frac{-15}{-5} = 3, \quad x_2 = \frac{-7 + x_3}{-2} = 2, \quad x_1 = \frac{5 - 3x_2 + x_3}{2} = \frac{5 - 3 \times 2 + 3}{2} = 1.$$

(本質的には、中学校で習う解法と同じ。)

付録: 記号・用語の説明 (1) ベクトルのノルム

$\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} \in \mathbb{R}^N$ に対して、ノルム $\|\boldsymbol{x}\|$ とは、次の性質を満たすものの こと。

(i) $\forall \boldsymbol{x} \in \mathbb{R}^N \ \|\boldsymbol{x}\| \geq 0$. 等号 $\Longleftrightarrow \boldsymbol{x} = 0$.

(ii) $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N \ \|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$

(iii) $\forall \boldsymbol{x} \in \mathbb{R}^N, \forall \lambda \in \mathbb{R} \ \|\lambda \boldsymbol{x}\| = |\lambda| \, \|\boldsymbol{x}\|$

よく使われるのは次式で定義する $\|\cdot\|_p$

$$\|\boldsymbol{x}\|_p := \left(\sum_{i=1}^{N} |x_i|^p \right)^{1/p} \qquad (1 \leq p < \infty),$$

$$\|\boldsymbol{x}\|_\infty = \max_{1 \leq i \leq N} |x_i| \, .$$

数学の授業では $\|\boldsymbol{x}\|_2$ を使うのが普通であるが、Wilkinson は $\|\boldsymbol{x}\|_\infty$ を良く使った。

付録: 記号・用語の説明 (2) 行列のノルム

$A = (a_{ij}) \in \mathbb{R}^{N\times N}$ に対して

$$\|A\| = \max_{\substack{\boldsymbol{x}\in\mathbb{R}^N \\ \boldsymbol{x}\neq 0}} \frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|}$$

を A の**作用素ノルム**と呼ぶ。
これは次の性質を満たす。

(i) $\|A\| \geq 0$. 等号 $\Longleftrightarrow A = O$.
(ii) $\|A + B\| \leq \|A\| + \|B\|$.
(iii) $\|\lambda A\| = |\lambda| \, \|A\|$
(iv) $\|AB\| \leq \|A\| \, \|B\|$, $\|A\boldsymbol{x}\| \leq \|A\| \, \|\boldsymbol{x}\|$
(v) 単位行列 I に対して $\|I\| = 1$.

ベクトルのノルムとして、$\|\boldsymbol{x}\|_p$ を用いたときの行列のノルムを $\|A\|_p$ と書くことにすると、$p = 1, \infty$ のときは比較的簡単で、

$$\|A\|_1 = \max_{1\leq j\leq N} \sum_{i=1}^{N} |a_{ij}|, \quad \|A\|_\infty = \max_{1\leq i\leq N} \sum_{j=1}^{N} |a_{ij}|.$$

参考文献表