区間解析†

柏木 雅英*

1 はじめに

数値計算において、計算を行うと同時にその結果の誤差評価をも同時に計算するような方法を総称して精度保証付き数値計算と呼び、近年急速な進歩を遂げている。精度保証付き数値計算の実現において最も基本的かつ重要な技法に、区間演算が挙げられる。区間演算とは、実数値を [下限, 上限] という2つの浮動小数点数で挟まれた区間で表現し、その区間同士の加減乗除等の演算を「演算結果として有り得る集合を包含するように」定義することにより行われるものである。

本稿では、区間演算を用いた精度保証付き数値計算の基本的な技法について解説する。

2 計算機における数値の表現

数値計算で用いられる数値(C言語でいうところの `double` や `float`)は、いわゆる浮動小数点形式で表されている。かつてはCPU毎に様々な方式が乱立していたが、IEEE 754 Std.の提案以降、この方式に統一されつつある。ここでは、この方式における数値の表現方法をおさらいしておく。

実数 x を2進数で表現したとき、0である場合を除いて、次のように正規化することが出来る：

$$x = 1.x_1x_2x_3\cdots \times 2^n$$

倍精度浮動小数点数(C言語における `double`)は、64bitで表され、その内訳は

符号 s (1bit)	指数部 e (11bit)	仮数部 m (52bit)
---------------	-----------------	-----------------

のとおりである。

s は符号 *bit* で、0なら正、1なら負である。

e は符号無し整数とみて0から2047までの値をとるが、通常の数(正規化数)では両端の値を除いた $1 \leq e \leq 2046$ であり、

$$x = \pm 1.m \times 2^{e-1023}$$

のように実数と対応する。このように、正規化された数は必ず先頭は1なのでそれはメモリには格納せず、52bitの領域で53bitの精度を達成している。この方法で表現できる絶対値最大の数は

$$1.111\cdots \times 2^{2046-1023} = 2^{1024} - 2^{971} \simeq 10^{308.25}$$

である。

また、この方法で表現できる絶対値最小の数は

$$1.000\cdots \times 2^{1-1023} = 2^{-1022} \simeq 10^{-307.65}$$

であるが、それより絶対値の小さい数は「非正規化数」として格納する。このときは $e=0$ とし、

$$x = \pm 0.m \times 2^{-1022}$$

のように実数と対応する。これによって、絶対値が $2^{-1074} \simeq 10^{-323.3}$ から $2^{-1022} - 2^{-1074}$ の間の数が表現できるが、非正規化数の場合は精度は53bitより少ないことに注意する必要がある。

0 は $e=m=0$ で表現する。

$e=2047$ は、 $\pm\infty$ ($m=0$ のとき)やNaNなどの特殊な数を表現するのに使用される。

倍精度浮動小数点の精度は、1より大きな最小の浮動小数点数と1との差(いわゆるマシンイプシロン)で見ると、 $2^{-52} \simeq 10^{-15.65}$ であり、およそ15桁の精度があると考えられる。ただしこれは数値が正規化数で表されている場合に限られる。

単精度浮動小数点数(C言語における `float`)の場合は、全体の長さが64bit $\rightarrow$ 32bit、指数部が11bit $\rightarrow$ 8bit、仮数部が52bit $\rightarrow$ 23bit、指数部のバイアス値が1023 $\rightarrow$ 127と変わっているだけで、基本的には同様である。

プロセッサによっては、倍精度浮動小数点数よりも高精度の拡張倍精度浮動小数点数を備える場合もあるが、これの表現形式は規定されておらず、プロセッサによって異なる。

倍精度浮動小数点を用いた場合、実数の内で計算機で正確に表現できる数は有限の高々 2^{64} 個に過ぎない(実際はもう少し少ない)。よって、例えば浮動小数点数を2つ加算した結果は、浮動小数点数で表せるとは限らず、それに近い浮動小数点数で近似することになる。これによって発生する誤差を、**丸め誤差**という。通常

† Interval Analysis
Masahide KASHIWAGI

* 早稲田大学理工学部情報学科
School of Science and Engineering, Waseda University

は最も近い浮動小数点数に丸めるが、後述するように IEEE 754 Std. に従う CPU では丸めの方法は変更可能である。

言うまでもないことだが、倍精度浮動小数点数の精度が10進数で15桁あるからと言って、計算結果が15桁正しいとは限らない。計算式と数値によってはわずかなステップの計算で有効数字が完全に失われることもめずらしくない。

3 区間演算

区間演算は、浮動小数点演算における丸め誤差の把握を行うための基本的な手法である。区間演算では、実数値を「下限, 上限」という2つの浮動小数点数で挟まれた区間で表現し、その区間同士の加減乗除等の演算を「演算結果として有り得る集合を包含するように」定義することにより行われる。例えば、区間 $X = [a, b]$, $Y = [c, d]$ に対する減算 $X - Y$ は、明らかに取り得る最小値は $a - d$ 、最大値は $b - c$ であるから、

$$X - Y = [a - d, b - c]$$

のように行う。いくつかの演算に対する区間演算をまとめておく¹。

$$X + Y = [a + c, b + d]$$

$$X - Y = [a - d, b - c]$$

$$X \times Y = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$$

$$X / Y = X \times \left[\frac{1}{d}, \frac{1}{c} \right] \quad ([c, d] \neq 0)$$

$$\exp(X) = [\exp(a), \exp(b)]$$

丸め誤差を伴う浮動小数点演算だけしか利用できないという仮定の下で数値計算の厳密な誤差評価を行う場合、次のような問題に直面することになる：

「ある計算の誤差の計算式が仮にあったとしても、その式を計算するときの誤差は誰が計算するのか？」

区間演算においても、両端の数を浮動小数点演算で計算せざるを得ない以上、この問題が生じる。区間演算では、区間の両端を計算する際に丸めの向きを「外向き」にしておくことによって丸め誤差の影響分を区間内に収めるという方法でこの問題を解決している。すなわち、区間の下限を計算するときには「切り捨て」、区間の上限を計算するときには「切り上げ」で計算を行う。こうすれば、厳密計算に比べて多少区間幅が大

きくなるものの、区間内に真値を包含しながら計算するという区間演算の性質は損なわれない。

例 3.1 簡単な例として、 $(1/3) \times 3$ を10進数で3桁の有効数字を持つシステムで計算することを考えよう。普通に計算すると、

$$\begin{aligned} 1/3 &= 0.333 \\ 0.333 \times 3 &= 0.999 \end{aligned}$$

となる。区間演算の場合は、最初に1と3を $[1, 1]$ 、 $[3, 3]$ のように幅0の区間としておき、計算を行う。

$$\begin{aligned} [1, 1] / [3, 3] &= [0.333, 0.334] \\ [0.333, 0.334] \times [3, 3] &= [0.999, 1.01] \end{aligned}$$

計算が進行すると、外向きの丸めのため数値は真値を含む微小区間となる。

このように、浮動小数点演算を用いた区間演算を実現するためには、確実に切り捨て(または切り上げ)で計算を行う必要がある。IEEE/754 Std.に従って設計されたCPUでは、加減乗除及び開平における丸めの発生の仕方を制御することが出来る。普段は真値に最も近い浮動小数点数に丸めるモードになっているが、これを「常に切り上げ」や「常に切り捨て」を行うように変更することが出来る。これを用いて、

1. 丸めモードを切り捨てに変更
2. 区間の下限を計算
3. 丸めモードを切り上げに変更
4. 区間の上限を計算
5. (必要なら丸めモードを通常の状態に戻す)

という手順で計算を行えば良い。ただし、丸めモードの変更が有効なのは加減乗除及び開平だけであり、expなどの超越関数の計算においては、

- ・ Taylor 展開などを用いて自力で計算し、厳密に誤差評価する。
- ・ 使用CPUの超越関数の誤差限界が分かる場合は、計算後に誤差分を補正する(下限の計算時は引き、上限の計算時は足す)。

などの方法を採らねばならない。

例3.1のように、幅0の区間から出発して区間演算を行うと丸め誤差の把握が可能である。一方、はじめから幅を持った区間から出発して区間演算を行えば、それは関数の値域を評価することになるのは区間演算の定義の仕方から自明であろう。すなわち、関数

$$f(x), f: \mathbf{R}^n \rightarrow \mathbf{R}^m$$

に対して、 $I \in \mathbf{IR}^n$ ($\mathbf{IR}$ は実数区間の集合、 $\mathbf{IR} = \{[a, b] \mid a \leq b, a, b \in \mathbf{R}\}$) から出発して区間演算を行った結果を $F(I) \in \mathbf{IR}^m$ とすると、

¹ここに挙げた乗除算の式は簡単に説明するためのもので、実際には a, b, c, d の正負によって場合分けすることで計算量を減らすのが普通である。

$$F(I) \supset \{f(x) | x \in I\}$$

を満たす。例えば、 $f(x) = x^2 + 2x$ に対して、区間 $I = [0, 1]$ として区間演算を行うと $[0, 3]$ を得るが、 $[0, 3] \supset \{f(x) | x \in [0, 1]\}$ である。

一般に区間を受け取り区間を返すような関数を区間写像という。また、 f に対して上の性質を満たすように作られた区間写像 $F: IR^n \rightarrow IR^m$ を、 f の区間包囲と言う。すなわち、関数の計算式が与えられている場合にその計算手順を区間演算に置き換えることは、関数の区間包囲を得る一つの手段である。

このように、区間演算には、

- ・丸め誤差の評価
- ・関数の値域の評価

という2つの役割がある。前者は無論精度保証付き数値計算の基礎となるものだが、方程式の解の精度保証などにおいてはむしろ後者が重要な役割を果たす。

4 区間演算の問題点とその解決

4.1 区間演算の問題点

区間演算の問題点の一つとして、確かに計算された区間は真の値を含むものの、区間幅が極端に広がってしまうことが多い点が挙げられる。これは特に関数の値域を評価する場合など、比較的幅の広い区間を扱った場合に顕在化する。例として、関数

$$f(x) = x^2 - 2x$$

の値域を、区間 $[0.9, 1.1]$ で評価する問題を考えよう。区間演算の定義に従って計算すると、

$$\begin{aligned} & [0.9, 1.1]^2 - 2 \times [0.9, 1.1] \\ &= [0.81, 1.21] - [1.8, 2.2] \\ &= [-1.39, -0.59] \end{aligned}$$

のようになるが、真の像は、

$$\{f(x) | x \in [0.9, 1.1]\} = [-1, -0.99]$$

であり、実に80倍もの区間幅の開きがあることが分かる。この現象は、2つの関数 x^2 と $2x$ が同じ x の関数で互いに相関があるにも関わらず、区間演算における減算ではそれを無視して独立な値として計算を行ってしまうことに原因がある。

なお、上の式は、 $x^2 - 2x = x(x - 2)$ と変型して評価すれば $[-1.21, -0.81]$ といくらか改善されるし、 $x^2 - 2x = (x - 1)^2 - 1$ とすれば真の像が得られる。この例で分かるように、区間演算においては数学的に同値な式でも計算の順序で大きく結果が異なる場合があり、注意が必要である。一般に、どのように式を変型すれば良

い結果が得られるかは極めて難しい問題であり、問題を多項式に限っても最適な方法は知られていない。

また、同様に変数間の相関を表現できないことに起因する問題点に、**Wrapping Effect** と呼ばれる現象が知られている。 n 次元ユークリッド空間 R^n ($n > 1$) の部分集合を区間で表す場合、区間ベクトルと呼ばれる各成分が区間であるようなもので表現するが、これでは R^n の超直方体領域しか表現することが出来ない。別の言葉で言えば、成分間の相関を表現できない。これによって、区間幅の増大が発生してしまうことがある。例えば、

$$A = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \quad B = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix},$$

$$I = \begin{pmatrix} [-1, 1] \\ [-1, 1] \end{pmatrix}$$

のとき、 $(AB)I$ と $A(BI)$ を区間演算で計算すると、

$$(AB)I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} I = \begin{pmatrix} [-1, 1] \\ [-1, 1] \end{pmatrix}$$

$$A(BI) = A \begin{pmatrix} [-1, 1] \\ [-1, 1] \end{pmatrix} = \begin{pmatrix} [-2, 2] \\ [-2, 2] \end{pmatrix}$$

のように、結果が大きく異なる。これは、 I に B を掛けた結果の真の像は超直方体領域でないので区間演算の結果としてはそれを包む(wrap する)超直方体となってしまう、領域の増大が発生し、またそれに A を掛けたときにも同様の現象が発生していることによる(図1参照)。

4.2 区間分割

4.1 節で述べた区間演算における過大評価の問題を解決する極めて自明な方法として、入力区間を分割して別々に評価し、最後に和集合をとる方法が考えられる。

すなわち、入力区間 I を

$$I = \bigcup_i I_i$$

のように分解し、それぞれについて区間演算を行なって最終結果を

$$\bigcup_i F(I_i)$$

とすればよい。この方法は単純で過大評価を抑える効果も高いが、当然 n 分割すれば n 倍時間がかかり、特に変数の数が多い問題に対しては十分に効果を発揮させるには分割数が大きくなりすぎるという問題がある。

図1: Wrapping Effectの様子

4.3 平均値形式

4.1 節で述べた区間演算における過大評価の問題を本質的に解決するための方法として変数間の相関性を考慮しつつ計算を行う方法が考えられる。平均値形式 (Mean Value Form) は、そのような方法で代表的なものの一つである。

関数

$$f(x), \quad f: \mathbf{R}^n \rightarrow \mathbf{R}^m$$

に対して、 $I \in \mathbf{IR}^n$ の像 $f(I) = \{f(x) | x \in I\}$ を評価することを考える。 $x, c \in I$ に対して、平均値の定理

$$f(x) - f(c) = \int_0^1 f'(c + t(x-c)) dt (x-c)$$

が成立する。ここで、

$$\int_0^1 f'(c + t(x-c)) dt \in \text{co}\{f'(x) | x \in I\}$$

(co は凸包) が成立することを利用すると、 $f(I) = \{f(x) | x \in I\}$ は

$$f(c) + F'(I)(I-c)$$

のように評価出来る。ここで F' は f の導関数 f' の区間包囲である。区間行列 $F'(I)$ は導関数 f' に区間 I を代入して区間演算を行うことによって計算することが出来る。区間行列は凸集合であるから、凸包を考える必要は無い。 c は入力区間 I に含まれる任意の点だが、普通は I の中心に取る。前述の例をこの方法で計算すると、

$$\begin{aligned} & f(c) + F'(I)(I-c) \\ &= f(1) + F'([0.9, 1.1])([0.9, 1.1] - 1) \\ &= -1 + [-0.2, 0.2] [-0.1, 0.1] \\ &= [-1.02, -0.98] \end{aligned}$$

とかなり改善されることが分かる。

4.1 節で述べたように x^2 と $2x$ の間の強い相関を無視したことが通常の区間演算での過大評価の原因であったが、ここでは

$$F'([0.9, 1.1]) = 2[0.9, 1.1] - 2 = [-0.2, 0.2]$$

のようにきちんと傾きがキャンセルされていることが分かる。

また、変数の数が多いときは実用的ではないが、一変数関数に関してはより高次の Taylor 展開を考える方法も役に立つ。Taylor 展開

$$f(x) = \sum_{k=0}^n \frac{1}{k!} f^{(k)}(c) (x-c)^k + R_{n+1}$$

に対して、Bernoulli の剰余項

$$R_{n+1} = \frac{1}{n!} \int_0^1 (1-t)^n f^{(n+1)}(c + t(x-c)) dt (x-c)^{(n+1)}$$

を考える ($f^{(k)}$ は f の k 階導関数)。ここで、 $s = (1-t)^{n+1}$ とおくと、

$$\begin{aligned} R_{n+1} &= \frac{1}{(n+1)!} \int_0^1 f^{(n+1)}(c + (1-n_{n+1}\sqrt{s})(x-c)) ds \cdot \\ &\quad (x-c)^{(n+1)} \end{aligned}$$

と変型できるので、平均値形式と同様に考えて、

$$\sum_{k=0}^n \frac{1}{k!} f^{(k)}(c) (I-c)^k + \frac{1}{(n+1)!} f^{(n+1)}(I) (I-c)^{n+1}$$

のような区間演算を行うことで $f(I)$ の包含が得られる。平均値形式は $n=0$ の場合に相当する。

4.4 自動微分法

上記の平均値形式を実際に用いるとき、 f の導関数 f' に区間を代入して微分(ヤコビ行列)を区間行列として計算する必要がある。 f が簡単で導関数が手で計算できるようなものならば問題はないが、長大なプログラムで f が記述されている場合など、それは困難であることが多い。数値的に微分を得る方法として $(f(x+\Delta x) - f(x))/\Delta x$ のような計算に基づく数値微分法がよく用いられるが、この方法では誤差の把握が出来ず、また区間値に対する微分を得ることは原理的に不可能である。

自動微分法は、関数 $f: \mathbf{R}^n \rightarrow \mathbf{R}$ に対して、その値の計算方法がプログラムの形で与えられたとき(すなわち与えられた値 x に対して $f(x)$ が数値計算出来るとき)、 $f(x)$ を数値的に計算する方法である。この方法では、「 f を数式的に手で微分してそれに x を代入する」という操作をすることなく、それと全く同じ解が得られる。

詳細は、[1] を見て頂くのが早いですが、ここで簡単に説明しておく。なお、以下で説明する方法は文献 [1] で言うところの **BU**(ボトムアップ) 算法と呼ばれるものに相当する。

BU 算法では、関数の計算中に現われる全ての変数において、通常の値に加えて入力変数 $x_1, \dots, x_n$ に関する偏微分値を保持する。すなわち、 n 変数関数の計算では全ての数値は

$$(v, \frac{\partial v}{\partial x_1}, \frac{\partial v}{\partial x_2}, \dots, \frac{\partial v}{\partial x_n})$$

のような $n+1$ 次元ベクトルで表現される。

まず、入力変数 $x_1, \dots, x_n$ の値 $v_1, \dots, v_n$ が与えられているとして、入力変数を表す計算機中の変数を

$$\begin{aligned} x_1 &= (v_1, 1, 0, 0, \dots, 0, 0) \\ x_2 &= (v_2, 0, 1, 0, \dots, 0, 0) \\ &\vdots \\ x_n &= (v_n, 0, 0, 0, \dots, 0, 1) \end{aligned}$$

の様に初期化する。

また、変数 p, q が $p = (p_0, p_1, \dots, p_n)$ 、 $q = (q_0, q_1, \dots, q_n)$ で表されていたとき、二項演算 $r = g(p, q)$ ($p+q$ 、 $p \times q$ など)や、単項演算 $r = h(p)$ ($\sin(p)$ など)は、次のように計算する：

$$\begin{aligned} r_0 &= g(p_0, q_0) \\ r_i &= \frac{\partial g}{\partial p}(p_0, q_0) p_i + \frac{\partial g}{\partial q}(p_0, q_0) q_i \quad (1 \leq i \leq n) \end{aligned}$$

$$r_0 = h(p_0)$$

$$r_i = \frac{dh}{dp}(p_0) p_i \quad (1 \leq i \leq n)$$

この規則に従って通常の関数の計算順序で計算を進めて行けば、最終結果(と副産物として中間変数)の入力変数に関する偏微分値が関数値と同時に得られるはずである。

平均値形式等に現われる導関数の区間包囲 $F'(I)$ の計算については、自動微分のための $n+1$ 次元ベクトルの各要素を全て区間とし、初期化時の $v_1, \dots, v_n$ を区間として計算を行えばよい。

5 非線形方程式の解の精度保証

5.1 基本的事項

区間演算または前節までに述べたその改良法によって、有限な計算手順によって表される数値計算法、いわゆる直接法については原理的には精度保証付きで行うことが出来る。例えば連立一次方程式は、Gauss の消去法の計算過程を区間演算に置き換えてやればその解を精度保証付きで求めることが出来る。ところが、直接解を表現することが出来ず、反復によって近似的に解を求める、いわゆる間接法については、区間演算化することには意味が無い。例えば、Newton 法では初期点 x_0 から出発して

$$x_{n+1} = x_n - f'(x_n)^{-1} f(x_n)$$

のような計算を行うが、この計算過程を区間演算化したところで、 $x_1, x_2, x_3, \dots$ のような途中の値に混入する丸め誤差を把握することは出来るが、肝心の真解との誤差は分からない。非線形方程式の場合はそもそも解が存在するかどうかすら分からないので、まず解の存在を保証することが必要で、次に誤差評価ということになる。

f が一変数関数の場合は、いわゆる2分法で解の存在を保証することが出来る。 f は連続とし、2点 a, b で f の値を(区間演算等を用いて)精度保証付きで計算する。このとき、 $f(a)f(b) < 0$ ならば、区間 $[a, b]$ に $f(x) = 0$ の解が存在することが保証される。区間幅が誤差評価であり、よく知られている方法で区間幅を狭めることも出来る。ところが、多変数関数の場合は、このような簡単な方法は知られておらず、別の方法が必要になる。

Newton 法の収束定理として Kantorovich の定理や占部の定理が知られているが、これらの成立条件が成立することを区間演算を用いて厳密に確かめることによって、真解との誤差評価を行うことが出来る。この方法は区間演算の難しい微分方程式等の解の精度保証

ではよく用いられる。ところが、有限次元の多変数非線形方程式

$$f(x)=0, \quad f: \mathbf{R}^n \rightarrow \mathbf{R}^n$$

の解の精度保証においては、2次導関数の評価をする必要がある、成分毎の評価を行わずノルムを用いているなどの理由により、余り性能が高いとは言えない。

Newton法の収束定理は関数解析の分野のいわゆる不動点定理に基づいて証明される。そこで、有限次元非線形方程式に対しては、Newton法の収束定理と同様に不動点定理に基づきながら、より直接的に区間演算を使ったシンプルな方法が用いられる。その代表的な方法が、次節で述べる **Krawczykの方法**である。

5.2 Krawczykの方法

まず準備として、不動点定理の内で最も基本的な縮小写像の原理を示す。

定理 5.1 (縮小写像の原理) X を $d(x, y)$ を距離とする完備距離空間とし、 $g: X \rightarrow X$ を全ての $x, y \in X$ に対して

$$d(g(x), g(y)) \leq \alpha d(x, y)$$

を満たすような写像とする。但し、 $\alpha < 1$ とする。(このような g を縮小写像と言い、実数 α を縮小定数と言う。) このとき、以下が成立する：

1. $g(x) = x$ を満たす点 x^* (不動点と言う) が X に唯一存在する。
2. x_0 を X 内の任意の点とし、点列 $\{x_k\}$ を

$$x_{k+1} = g(x_k)$$

で定めると、 x_k は x^* に収束し、その収束速度は

$$d(x_k, x^*) \leq \frac{\alpha^k}{1-\alpha} d(x_1, x_0)$$

で表される。

3. X 内の任意の点 x_0 に対して、

$$d(x_0, x^*) \leq \frac{d(g(x_0), x_0)}{1-\alpha}$$

上の定理は一般的な形で書いてあるが、本稿では n 次元ユークリッド空間 $\mathbf{R}^n$ を扱っているので、

X : $\mathbf{R}^n$ 内の閉集合

$d(x, y)$: $\|x - y\|$ 。但しノルムは

$\|x\| = \max |x_i|$ (最大値ノルム) で定める。

のように読み替えれば良い。

非線形方程式

$$f(x)=0, \quad f: \mathbf{R}^n \rightarrow \mathbf{R}^n$$

の解の存在を保証するために、これを同値な不動点形式に変換し、不動点定理を適用することを考える。単純に

$$x = x + f(x)$$

のようにしても良いが、これでは右辺の縮小性が成立するかどうかは f によることになってしまう。近似解の近辺で縮小性を成立させるには、傾きを 0 に近づければ良い。そこで、 $L: \mathbf{R}^n \rightarrow \mathbf{R}^n$ を近似解の近辺での f の傾きを近似する行列とし、これを用いて f の傾きを補正して、

$$x = x - L^{-1}f(x)$$

のようにする ($f(x) = 0$ を $-L^{-1}f(x) = 0$ だっただと考え)。以下この右辺を $g(x)$ とおく：

$$g(x) = x - L^{-1}f(x)$$

L が逆行列を持つならば x が $f(x) = 0$ を満たすことと x が $x = g(x)$ を満たすことは同値なので、以下 $x = g(x)$ の不動点の存在を示すことを考えれば良い。以下、区間 $I \subset \mathbf{R}^n$ とし、 I に g の不動点が存在することを示すことを考える。示すべきことは、

1. $\{g(x) \mid x \in I\} \subset I$ (g が I から I への写像であること)
2. g が I 上で縮小写像であること。

の2つである。

前者は、区間演算を用いて

$$G(I) = I - L^{-1}F(I)$$

のように g に対する区間演算を用い、 $G(I) \subset I$ を確認すれば容易に確認できるように見える。しかし、区間演算の減算の定義の仕方では、 $c = a - b$ ならば c の区間幅は a の区間幅と b の区間幅の和となることを考えれば、決して $G(I) \subset I$ は成立しない！写像 $g(x) = x - L^{-1}f(x)$ は、傾きを 0 に近づけるためによく似た傾きの関数の減算を行なっている訳で、これは4.1節で述べた区間演算の苦手な計算の形そのものである。よって、区間演算の過大評価を抑えるための何らかの手法を併用する必要がある。Krawczykの方法では、平均値形式を用いて、 c を I の中心として

$$\begin{aligned} K(I) &= g(c) + G'(I)(I - c) \\ &= c - L^{-1}f(c) + (E - L^{-1}F'(I))(I - c) \end{aligned}$$

(E は単位行列) のような **Krawczyk** 写像 K について、

$$K(I) \subset I$$

を確認する。

後者の条件については、次のように確認する。平均

値の定理により、 $x, y \in I$ ならば

$$\|g(x) - g(y)\| \leq \max_{x \in I} \|g'(x)\| \|x - y\|$$

が成立する。ここで行列 $g'(x)$ のノルムは、

$$\|A\| = \sup_{\|x\|=1} \|Ax\|$$

となるように定義されたもので、 $\mathbf{R}^n$ のノルムが $\max |x_i|$ (最大値ノルム) ならば

$$\|A\| = \max_i \sum_j |A_{ij}|$$

で計算できる。すなわち、区間行列のノルムを

$$\|A\| = \max_i \sum_j \max_{x \in A_{ij}} |x|$$

で定義し、

$$\|G'(I)\| = \|E - L^{-1}F'(I)\| < 1$$

が成立すれば、 g は I 上で縮小写像であることが言える。

以上をまとめると、次のようになる：

定理 5.2 (Krawczyk の方法) $f: \mathbf{R}^n \rightarrow \mathbf{R}^n$ 、 $I \in \mathbf{IR}^n$ 、 c を I の中心、 $L \simeq f'(c)$ を正則な行列、 F' を f の区間包囲とする。

$$K(I) = c - L^{-1}f(c) + (E - L^{-1}F'(I))(I - c)$$

としたとき、

$$K(I) \subset I$$

$$\|E - L^{-1}F'(I)\| < 1$$

が成立するならば、以下が成立する：

1. 方程式 $f(x) = 0$ の解 x^* は I に唯一存在する。
2. x_0 を I 内の任意の点とし、点列 $\{x_k\}$ を

$$x_{k+1} = x_k - L^{-1}f(x_k)$$

で定めると、 x_k は x^* に収束し、その収束速度は

$$\|x_k - x^*\| \leq \frac{\alpha^n}{1 - \alpha} \|L^{-1}f(x_0)\|$$

で表される。但し $\alpha = \|E - L^{-1}F'(I)\|$ である。

このように、Krawczyk の方法で解の存在が示されれば、同時に Newton 法の収束も保証される。ただし、この定理は数学的なものであり、Newton 法に混入する丸め誤差は考慮されていないことに注意する必要がある。また、 $f(c)$ の計算にも丸め誤差が混入するため、この計算は実際に区間演算で行なう（従って $f(c)$ は幅の余り大きくない区間になる）必要がある。

例として、

$$f(x) = f \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 + x_2^2 - 1 \\ x_1 - x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

の解が、

$$I = \begin{pmatrix} [0.6, 0.8] \\ [0.6, 0.8] \end{pmatrix}$$

に存在することを示そう。

$$c = \begin{pmatrix} 0.7 \\ 0.7 \end{pmatrix}, f'(x) = \begin{pmatrix} 2x_1 & 2x_2 \\ 1 & -1 \end{pmatrix},$$

$$f'(c)^{-1} \begin{pmatrix} \frac{5}{14} & \frac{1}{2} \\ \frac{5}{14} & -\frac{1}{2} \end{pmatrix}$$

より、 $f'(c)^{-1}$ の近似として L^{-1} を、

$$L^{-1} = \begin{pmatrix} 0.4 & 0.5 \\ 0.4 & -0.5 \end{pmatrix}$$

と定める。このとき、

$$E - L^{-1}F'(I)$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0.4 & 0.5 \\ 0.4 & -0.5 \end{pmatrix} \begin{pmatrix} [1.2, 1.6] & [1.2, 1.6] \\ 1 & -1 \end{pmatrix}$$

$$= \begin{pmatrix} [-0.14, 0.02] & [-0.14, 0.02] \\ [-0.14, 0.02] & [-0.14, 0.02] \end{pmatrix}$$

であるから、

$$\|E - L^{-1}F'(I)\| = 0.28 < 1$$

であり、縮小性は確認できた。また、

$$K(I) = c - L^{-1}f(c) + (E - L^{-1}F'(I))(I - c)$$

$$= \begin{pmatrix} 0.7 \\ 0.7 \end{pmatrix} - \begin{pmatrix} 0.4 & 0.5 \\ 0.4 & -0.5 \end{pmatrix} \begin{pmatrix} -0.02 \\ 0 \end{pmatrix}$$

$$+ \begin{pmatrix} [-0.14, 0.02] & [-0.14, 0.02] \\ [-0.14, 0.02] & [-0.14, 0.02] \end{pmatrix} \begin{pmatrix} [-0.1, 0.1] \\ [-0.1, 0.1] \end{pmatrix}$$

$$= \begin{pmatrix} [0.68, 0.736] \\ [0.68, 0.736] \end{pmatrix}$$

より $K(I) \subset I$ であり、よって $f(x) = 0$ の解は I に唯一存在することが保証された。

5.3 近似解の誤差評価

前節で説明した Krawczyk の方法により、非線形方程式の解の存在を保証することが出来る。しかし、最もよくある状況と思われる、近似解が与えられてその近くに真の解があることを保証したい場合、近似解を元に Krawczyk の方法を適用するための区間 I を設定する必要がある。区間が大きすぎれば区間内の関数の変動が大きくなり縮小性が成立しなくなる。しかし区間が小さすぎれば真の解を含まなくなってしまう。

以下に、区間の決定法の一例を示す。近似解 c が与えられたとき、区間 I を

$$I = \rho \|f'(c)^{-1}f(c)\| \begin{bmatrix} [-1, 1] \\ [-1, 1] \\ \vdots \\ [-1, 1] \end{bmatrix}$$

で決定する。但し、 ρ は $\rho > 1$ なる定数である。このように区間を決定したとき、適切な条件の下で、充分良い近似解において Krawczyk の方法は成功する [2]。なお、理由は省略するが、 $\rho = 2$ とするのが最適であると考えられる。

6 おわりに

本稿では、区間演算を用いた精度保証付き数値計算の基本的な技法について解説した。更に勉強するためには、[3] [4] [5] [6] などの文献を参照されたい。

参 考 文 献

- [1] 伊理正夫, 久保田光一: “高速自動微分法(I), (II)”, 応用数理, Vol.1, No.1, No.2, pp.17-35, pp.53-63, 1991.
- [2] 柏木 雅英, 大石 進一: “区間解析と有理数演算による非線形方程式の近似解の精度保証”, 電子情報通信学会論文誌(A), Vol.J77-A, No.10, pp.1372-1382, 1994.
- [3] 特集「精度保証付き数値計算とその応用」, 情報処理, Vol.31, No.9, pp. 1176-1234, 1990.
- [4] 大石 進一: “非線形解析入門”, コロナ社, 1997 (ISBN4-339-02600-X).
- [5] 中尾 充宏, 山本 野人: “精度保証付き数値計算 コンピュータによる無限への挑戦”, 日本評論社, 1998 (ISBN4-535-78258-X).
- [6] 大石 進一: “数値計算”, 裳華房, 1999 (ISBN4-7853-1514-8).

(2003年3月5日 受付)

[問合せ先]

〒169-8555 新宿区大久保3-4-1

電子情報通信学会正員

柏木 雅英

TEL: 03-5286-3484

FAX: 03-5286-3484

E-mail: kashi@waseda.jp

著 者 略 歴

かしわぎ まさひで
柏木 雅英 [電子情報通信学会正員]

1989早稲田大学理工学部電子通信学科卒。
1991同大学院理工学研究科修士課程修了。
1994同大学院理工学研究科博士課程修了。
1993同大理工学部助手に着任。
1995九州大学大学院システム情報科学研究科助教授。
1998より早稲田大学勤務。情報学科助教授。
精度保証付き数値計算に関する研究に従事。
1991年度、1997年度論文賞受賞。