2001 年度情報処理 II　第 10 回

数学のためのコンピューター (1) 方程式の数値解法

桂田 祐史（かつらだ まさし）

2001 年 6 月 28 日

コンピューターで数値計算をして (有限次元の) 方程式を解く方法について学ぶ[1]。厳密解を求めることにすると、線形方程式以外は例外的な状況をのぞいて解けない[2]。しかし、有限精度の解 (近似解) で満足することにすれば、かなり多くの方程式が解けることになる。

ここでは二分法と Newton（ニュートン） 法を取り上げるが、これらは解析学の学習とも関係が深い。二分法は中間値の定理の区間縮小法による証明 (中間値の定理はいわゆる「存在定理」であるが、この証明は「構成的な」証明であると言える) そのものであると考えられよう。また Newton 法は陰関数の定理や逆関数の定理の証明に用いることもできるし、実際に陰関数・逆関数の計算に利用できる。

1　はじめに

今回は次の例題を考える。

例題 1:　二分法によって、方程式 $\cos x - x = 0$ の解を計算せよ。

例題 2:　Newton 法によって、方程式 $\cos x - x = 0$ の解を計算せよ。

「方程式を解け」という問題はしばしば現れる。基本的で大事な方程式はその性質を学んできたが、それ以外にも色々な方程式がある。方程式は解が存在しても、紙と鉛筆の計算で具体的に解くのが難しいことがしばしばある[3]。この例題の方程式もそういうものの一つで、解がただ一つあることは簡単に分かるが (後の補足を参照)、その解を簡単な式変形等で求めることは出来そうにない。これに計算機でチャレンジしよう、というのが今週の例題である。

計算機で方程式を扱う場合には、計算機ならではのやり方がある。有限回の計算で真の解 (無限精度の解) を求めることをあきらめて、真の解を求めるには無限回の演算が必要だが、有限桁の要求精度を持つ解 (近似解) はそこそこの回数の基本的な演算 (四則や初等関数の計算) で求まるような方法 — 近似解法 — を採用する、というものである。従ってアルゴリズムは、大抵繰り返しのあるものになる。

[1] 方程式を難しくする「原因」として、非線型性と無限次元性がある。ここでは非線型性を取り上げる。

[2]「例外的な状況」は重要でないと勘違いしないように。解けるような例外的な問題には重要なものも多い。

[3] 方程式によっては、人間の手計算では実際的な解法がないものもある、というかそういうものの方が多いわけだが、大学二年次までの段階では、具体的に解ける問題を扱うことの方が多いので、ピンと来ないかもしれない。

ここで解説する近似解法は、適用できる範囲はかなり広く、計算機を使って計算することになる人は、今後も何度も「お世話になる」はずである。

2　方程式の分類

方程式といっても色々なものがあるが、ここでは微分や積分を含まない、有限次元の、「普通の」もの、つまり既知関数 $f : \mathbf{R}^n \supset \Omega \longrightarrow \mathbf{R}^m$ を用いて表される、未知数 x についての方程式

$$f(x) = 0$$

について考える。

2.1　線形方程式 — 比較的簡単

f が x の 1 次式である場合、方程式を線型方程式と呼ぶ。これは、 f が適当な行列 $A \in M(m, n; \mathbf{R})$, ベクトル $b \in \mathbf{R}^m$ を用いて $f(x) = Ax - b$ と表されるということで、いわゆる (連立) 1 次方程式になる。この場合は有限回の四則演算で解が求まる。(良く知っているように) A が n 次正則行列であった場合は $x = A^{-1}b$. 既に何らかの解法[4]を習ったことがあるはずである。この問題はみかけよりも奥が深く、また非常に応用範囲が広いので、実に精力的に研究されていて、面白い手法も少なくないが、この講義では紹介を見送る。

研究課題 10-1

連立 1 次方程式を解くための共役勾配法(きょうやくこうばいほう) (CG method) について調べ、プログラムを書いて実験せよ。

2.2　非線形方程式 — なかなか難しい

ここでは非線形方程式について考えよう。もっとも簡単な非線形方程式は、 2 次以上の代数方程式

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0 = 0 \quad (a_n \neq 0)$$

であろう。「n 次代数方程式は n 個の根を持つ」ことは常識として知っているはず。また、次数 n が 2 の場合は「2 次方程式」で、根の公式は中学校で学んでいる (もうすぐ消える？)。さらに n が 3, 4 である場合も、(2 次方程式ほどポピュラーではないが) 根の公式[5]がある。ところが、「n が 5 以上の場合は、四則とべき根のみを有限回用いた根の公式は存在しない」ことが Galois(ガロア) 理論を用いて証明されている (3 年の代数学で習うはず)。

比較的簡単なはずの代数方程式でもこんな調子なのだから、より一般の非線形方程式を簡単な式変形のみで解くことは、よほど運が良くない限り駄目だ、ということになる。

研究課題 10-2

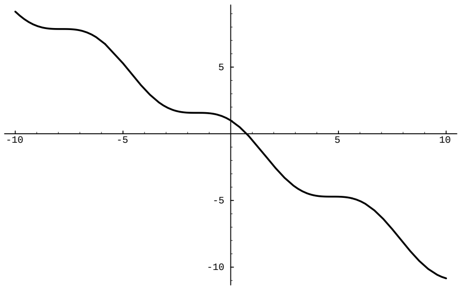
3 次代数方程式を解くための Cardano(カルダノ) の方法について調べ、プログラムを書いて実験せよ。

[4] 例えば掃き出し法、 Gauss の消去法など。理論的には Cramer の方法 (これはあまり実用的でない)。

[5] もっとも、とても複雑で、紙と鉛筆で計算するのは (少なくとも私は) うんざりしてしまう。

3 非線形方程式を計算機で解く

ここでは例題の方程式 $\cos x - x = 0$ について考えよう。$f(x) = \cos x - x$ とおいて、f を少し調べてみれば (このすぐ後に述べる)、この方程式にはただ一つの解があって、それは区間 $(0,1)$ にあることが分かる。大雑把に言うと、グラフの概形は $y = -x$ をサイン・カーブ風に波打たせたものになる。この唯一の解を求めることが目標である。



方程式が区間 $(0,1)$ にただ一つの解を持つことの証明

まず $f'(x) = -\sin x - 1 \le 0$ $(x \in \mathbf{R})$ で、特に $x = \pi/2 + 2n\pi$ $(n \in \mathbf{Z})$ 以外のところでは $f'(x) < 0$ であるから、f は狭義の単調減少関数である。そして $f(0) = 1 > 0$, $f(1) = \cos 1 - 1 < 0$ ゆえ、方程式 $f(x) = 0$ は区間 $(0,1)$ 内に少なくとも一つの解を持つが (中間値の定理)、f の単調性からそれは $\mathbf{R}$ 全体でただ一つの解であることが分かる。■

3.1 二分法 (bisection method)

微積分で基本的な中間値の定理を復習しよう。

定理 3.1 (中間値の定理) $f : [\alpha, \beta] \to \mathbf{R}$ を連続関数、$f(\alpha)f(\beta) < 0$ とすると、$f(c) = 0$ となる $c \in (\alpha, \beta)$ が存在する。

(つまり $f(\alpha)f(\beta) < 0$ となる α, β があれば、方程式 $f(x) = 0$ の解 $x = c$ が区間 (α, β) 内に存在するということ。)

この定理の証明の仕方は色々あるが、代表的なものに区間縮小法を使ったものがある。それは以下のような筋書きで進む。

次の手順で帰納的に数列 $\{a_n\}$, $\{b_n\}$ を定める。

(i) $a_0 = \alpha$, $b_0 = \beta$ とする。

(ii) 第 n 項 a_n, b_n まで定まったとして、$c_n = (a_n + b_n)/2$ とおき、$f(a_n)f(c_n) < 0$ なら $a_{n+1} = a_n$, $b_{n+1} = c_n$, そうでないなら $a_{n+1} = c_n$, $b_{n+1} = b_n$ とする。

すると、

$$a_0 \le a_1 \le a_2 \le \cdots \le a_n \le a_{n+1} \le \cdots, \quad \cdots \le b_{n+1} \le b_n \le \cdots \le b_2 \le b_1 \le b_0$$

$$a_n < b_n \le b_0 \quad (n \in \mathbf{N}) \quad \text{さらに} \quad a_0 \le a_n < b_n \quad (n \in \mathbf{N}),$$

$$b_n - a_n = (\beta - \alpha)/2^n \to 0 \quad (\text{as } n \to \infty),$$

$$f(a_n)f(b_n) \le 0 \quad (n \in \mathbf{N}).$$

これから

$$\lim_{n\to+\infty} a_n = \lim_{n\to+\infty} b_n = c, \quad \alpha < c < \beta$$

と収束して

$$f(c) = 0$$

が成り立つことが分かる。■

以上の証明の手続きから、 $f(\alpha)f(\beta) < 0$ となる α, β が分かっている場合に、方程式 $f(x) = 0$ の近似解を求めるアルゴリズムが得られます (以下では $\leftarrow$ は変数への代入を表します)。

二分法のアルゴリズム

(1) 目標とする誤差 ε を決める。

(2) $a \leftarrow \alpha$, $b \leftarrow \beta$ とする。

(3) $c \leftarrow (b+a)/2$ として $f(a)f(c) < 0$ ならば $b \leftarrow c$、そうでなければ $a \leftarrow c$ とする

(4) $|b-a| \ge \varepsilon$ ならば (1) に戻る。そうでなければ c を解として出力する。

注意 3.1 (反復の停止のための ε) 目標とする誤差としては、C 言語で倍精度浮動小数点数 `double` (実習で利用している C コンパイラーでは相対精度が 16 桁弱) を用いる場合は解の絶対値の推定値 (本当に大雑把な、桁数の目安がつく位のもので構わない) の大きさに 10^{-15} をかけた数程度にするのが適当です[6]。それ以上小さく取っても、使用している浮動小数点数の体系の能力を越えてしまうことになるので意味がない。この問題の場合は解は区間 $(0,1)$ の真ん中位にあるので、ざっと 1 程度ということで、 $\varepsilon = 10^{-15}$ 位が良いだろう (この数を表すのに、C 言語では "`1.0e-15`" と書く)。

3.2 Newton 法

非線形方程式を解くためのもう一つの代表的な方法が **Newton** 法である。

これは f が微分可能な関数で、方程式 $f(x) = 0$ の近似解 x_0 が得られている時、漸化式

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (n = 0, 1, 2, \cdots)$$

で数列 $\{x_n\}_{n=0,1,2,\cdots}$ を定めると、適当な条件[7]の下で

$$\lim_{n\to+\infty} x_n = x_*$$

と収束し、極限 x_* は方程式の解になっている:

$$f(x_*) = 0$$

[6] 単精度の場合には 10^{-7} 程度にすべきであろう。

[7] Newton 法が収束するための十分条件は色々知られているが、ここでは説明しない。簡単なものは微分積分学のテキストに載っていることも多い。

ということを利用したもので、実際のアルゴリズムは次のようになる。

Newton 法のアルゴリズム

(1) 適当な初期値 x_0 を選ぶ。

(2) $x \leftarrow x_0$

(3) $x \leftarrow x - f(x)/f'(x)$ とする。

(4) まだ近似の程度が十分でないと判断されたら (3) に戻る。そうでなければ x を解として出力する。

3.3 二分法 vs. Newton 法

ここで紹介した二つの方法はどちらが優れているだろうか？それぞれの長所・短所をあげて比較してみよう。

二分法 — f が微分可能でなくとも連続でありさえすれば適用できる。しかし f は 1 変数実数値関数でない場合は適用が難しい (特に実数値であることはほとんど必要であると言ってよい)。$f(\alpha)f(\beta) < 0$ なる α, β が見つかっていれば、確実に解が求まるが、収束はあまり速くない。1 回の反復で 2 進法にして 1 桁ずつ精度が改善されていく程度である。

Newton 法 — 適用するには少なくとも f が微分可能である必要がある。微分可能であっても f の実際の計算が難しい場合もあるので、そういう場合も適用困難になる。一方 f は多変数ベクトル値関数でも構わない (それどころか無限次元の方程式にも使うことが出来る)。適切な初期値を探すことは、場合によってはかなり難しい。求める解が重解でない場合には、十分真の解に近い初期値から出発すれば 2 次の収束となり (合っている桁数が反復が一段進むごとに 2 倍になる)、非常に速い。

総合的に見て「まずは Newton 法を使うことを考えよ、それが困難ならば二分法も考えてみよ。」というところでしょうか。

4 レポート課題 6

以下から二つ以上 (好きなだけ) 解いて、提出して下さい。〆切は 7 月 11 日。

課題 6-1 色々な方程式を二分法、Newton 法で解いてみよ。初期値の選び方を変えて、収束するかしないか、試しなさい。収束の判定条件には注意を払うこと。最終的に得られる精度や、必要な反復の回数はどの程度になるか。Newton 法の収束のための十分条件を本などで探すことができたら、それも説明すること。

ここでは解説しないが、初等関数なども計算機の中では、四則演算などの簡単な演算の組合せで計算されていることが多い。

課題 6-2　平方根 $\sqrt{a}$ や立法根 $a^{1/3}$ を方程式の解の形に定式化して、 Newton 法で解いて見よ。その結果を C 言語のライブラリィ関数 `sqrt()` や `pow()` [8] で計算した値と比較してみよ。

特に逆関数の計算にも使える (y が与えられているとして方程式 $f(x) = y$ を x について解けば、 $x = f^{-1}(y)$ が得られるはず)。

課題 6-3　C 言語のライブラリィ関数 `asin()`, `acos()`, `atan()` は使わずに、 arcsin, arccos, arctan を計算するプログラムを作り、実験せよ。

Newton 法の意味

Newton 法の式の意味を簡単に説明しよう。微分の定義によると、 x が a に十分近いところでは、 f は「接線の式」で近似されることが期待される:

$$f(x) \fallingdotseq f'(a)(x-a) + f(a).$$

今 a が $f(x) = 0$ の解に十分近いとすると、 $f(x) = 0$ の代わりに

$$f'(a)(x-a) + f(a) = 0$$

を解くことにより、 a よりも精度の高い近似解が得られると考えるのは自然であろう。実際に実行すると、まず移項して

$$f'(a)(x-a) = -f(a).$$

両辺に $[f'(a)]^{-1}$ をかけて

$$x - a = -[f'(a)]^{-1}f(a),$$

ゆえに

$$x = a - [f'(a)]^{-1}f(a) = a - \frac{f(a).}{f'(a)}.$$

多変数の場合も、 $[f'(a)]^{-1}$ を Jacobi 行列の逆行列と考えれば、まったく同様に Newton 法が使える。

課題 6-4　連立方程式

$$\begin{aligned} x^2 - y^2 + x + 1 &= 0 \\ 2xy + y &= 0 \end{aligned}$$

を Newton 法を用いて解くプログラムを作れ。ヒント:

$$\vec{x} = \begin{pmatrix} x \\ y \end{pmatrix},$$

$$f(\vec{x}) = \begin{pmatrix} x^2 - y^2 + x + 1 \\ 2xy + y \end{pmatrix}$$

とおくと、方程式は $f(\vec{x}) = 0$ と書ける。 f の $\vec{x}$ における Jacobi 行列を $f'(\vec{x})$ とすると、 Newton 法の式は

$$\vec{x}_{n+1} = \vec{x}_n - [f'(\vec{x}_n)]^{-1} f(\vec{x}_n)$$

となる。初期値 $\vec{x}_0$ を $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ として実験せよ。

---

[8] `pow(a,b)` で a の b 乗が計算できる。例えば `pow(1.0, 1.0/3.0)` で a の立法根が計算できる。

5 例題を解くプログラム例

二分法のプログラム bisection.c, newton.c は情報処理 II の WWW ページから入手できる。 netscape の「ファイル・メニュー」の「名前をつけて保存」を用いてセーブする。

5.1 Newton 法の場合

1 を初期値として、許容精度 10^{-15} を指示して Newton 法で解かせたのが以下の結果 (入力は 1 1e-15)。

```
waltz21% cc -o newton newton.c -lm
waltz21% ./newton
 初期値 x0, 許容精度ε=1 1e-15
f(  0.7503638678402439)=-1.89e-02
f(  0.7391128909113617)=-4.65e-05
f(  0.7390851333852840)=-2.85e-10
f(  0.7390851332151607)= 0.00e+00
f(  0.7390851332151607)= 0.00e+00
waltz21%
```

注意 5.1 Newton 法の繰り返しを停止させるための良いルールを独力で発見するのはかなり難しい。上の例題プログラムの採用したルールは、多くのプログラムで採用されているやり方ではあるが、いつでもうまく行く方法とは言えない。現時点で「これが良い方法」と見なされている方法は一応あるが、結構複雑なのでここでは説明しない。

5.2 二分法の場合

区間 $(0, 1)$ 内に解があることがわかるから、二分法で許容精度 10^{-15} を指示して解かせたのが以下の結果 (入力は 0 1 1e-15)。関数値が、区間の左端では正、右端では負になったまま区間が縮小して行くのを理解しよう。

```
waltz21% cc -o bisection bisection.c -lm
waltz21% ./bisection
 探す区間の左端α, 右端β, 許容精度ε=0 1 1e-15
f(  0.5000000000000000)= 3.78e-01, f(  1.0000000000000000)=-4.60e-01
f(  0.5000000000000000)= 3.78e-01, f(  0.7500000000000000)=-1.83e-02
f(  0.6250000000000000)= 1.86e-01, f(  0.7500000000000000)=-1.83e-02
f(  0.6875000000000000)= 8.53e-02, f(  0.7500000000000000)=-1.83e-02
f(  0.7187500000000000)= 3.39e-02, f(  0.7500000000000000)=-1.83e-02
f(  0.7343750000000000)= 7.87e-03, f(  0.7500000000000000)=-1.83e-02
f(  0.7343750000000000)= 7.87e-03, f(  0.7421875000000000)=-5.20e-03
f(  0.7382812500000000)= 1.35e-03, f(  0.7421875000000000)=-5.20e-03
中略
f(  0.7390851332151556)= 8.55e-15, f(  0.7390851332151627)=-3.44e-15
f(  0.7390851332151591)= 2.55e-15, f(  0.7390851332151627)=-3.44e-15
f(  0.7390851332151591)= 2.55e-15, f(  0.7390851332151609)=-4.44e-16
f(  0.7390851332151600)= 1.11e-15, f(  0.7390851332151609)=-4.44e-16
f(  0.7390851332151600)= 1.11e-15
waltz21%
```