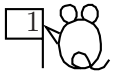
数値計算講義　第5 回

2 分法とニュートン法
– 非線型方程式の一般解法 –

金子　晃

kanenko@mbk.nifty.com
http://kanenko.a.la9.jp/

数学で解けない方程式を計算機で解きたい！



例1 ： $\cos x = x$

$[0, \frac{\pi}{2}]$ に解が唯一つ有ることは，グラフ (中間値定理) から分かる．

区間の両端で連続函数の符号が異なっていれば, 中に必ず零点が有る!

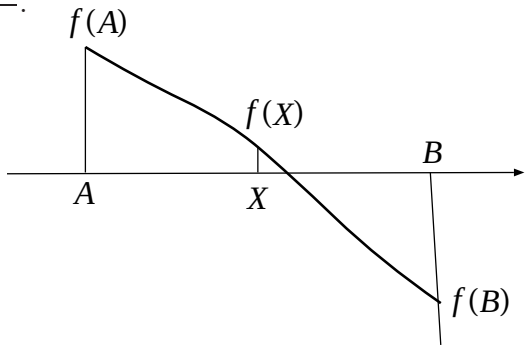
例2 ： $x^5 - x + 1 = 0$

Gauss によれば，根は5 個有る．奇数次なら, 少なくとも1 個は実根.

最も汎用的な方法は2 分法である！

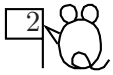
区間を半分にし, 符号変化がある方を残すという操作を繰り返す.

n 回反復したときの誤差は $\frac{B-A}{2^n}$.



まず，例1 で実験してみよう．

FORTRAN プログラム num5-1.f

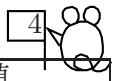



```
      PROGRAM BISECT
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      F(X)=DCOS(X)-X      ! 文函数（いわゆるマクロ）
      A=0.0D0
      B=1.57D0            ! π/2 の近似値はこのくらい粗くても十分
      FA=F(A)             ! 使うのは符号だけ
      FB=F(B)             ! 同上
      IF (FA*FB.GT.0.0D0) THEN
         WRITE(*,*) 'WRONG INTERVAL'
         STOP
      END IF
      DO I=1,10000    ! 反復回数は適当に設定
         X=(A+B)/2
         FX=F(X)
         WRITE(*,100) I,': H= ',B-X,', X= ',X,', ERROR: ',FX
         IF (B-A .LT. 0.2D-15) STOP   !誤差<0.2x10^(-15)で停止
         IF (FX*FA .LT. 0.0D0) THEN  !符号を見て次はどちらか決める
            B=X                      !このとき FX は FB と同符号
         ELSE
            A=X                      !このとき FX は FA と同符号
         END IF
      END DO       ! DO ループの終点を行番号で指定しない書き方
  100 FORMAT(1H ,I2,A5,F18.15,A5,F18.15,A9,F18.15)
      END
```

.LT. は不等号 < のこと. 昔は計算機で使える記号が極端に
少なかったのでこのように書いた.

	区間幅	中点の座標	函数値
1	0.785000000000000	0.785000000000000	-0.077611730832800
2	0.392500000000000	0.392500000000000	0.531455699470272
3	0.196250000000000	0.588750000000000	0.242885481025376
4	0.098125000000000	0.686875000000000	0.086356424607338
5	0.049062500000000	0.735937500000000	0.005264252036190
6	0.024531250000000	0.760468750000000	-0.035955750807525
7	0.012265625000000	0.748203125000000	-0.015290618361843
8	0.006132812500000	0.742070312500000	-0.004999322074341
9	0.003066406250000	0.739003906250000	0.000135939987751
10	0.001533203125000	0.740537109375000	-0.002430823505879
11	0.000766601562500	0.739770507812500	-0.001147224722764
12	0.000383300781250	0.739387207031250	-0.000505588089452
13	0.000191650390625	0.739195556640625	-0.000184810478965
14	0.000095825195312	0.739099731445312	-0.000024431852339
15	0.000047912597656	0.739051818847656	0.000055754916060
16	0.000023956298828	0.739075775146484	0.000015661743944
17	0.000011978149414	0.739087753295898	-0.000004385001177
18	0.000005989074707	0.739081764221191	0.000005638384639
19	0.000002994537354	0.739084758758545	0.000000626695045
20	0.000001497268677	0.739086256027222	-0.000001879152238
21	0.000000748634338	0.739085507392883	-0.000000626228389
22	0.000000374317169	0.739085133075714	0.000000000233380

途中を省略してループの最後のところを示す：



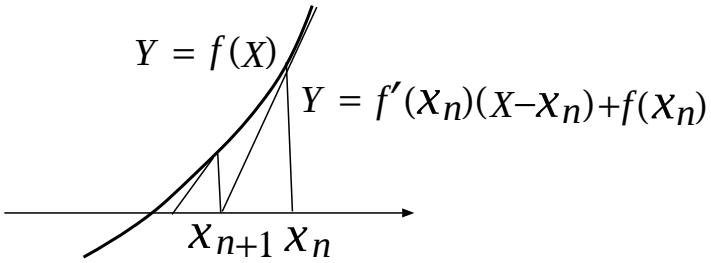
回数	区間幅	中点の座標	函数値
46	0.000000000000022	0.739085133215180	-0.000000000000033
47	0.000000000000011	0.739085133215169	-0.000000000000014
48	0.000000000000006	0.739085133215164	-0.000000000000005
49	0.000000000000003	0.739085133215161	-0.000000000000000
50	0.000000000000001	0.739085133215159	0.000000000000002
51	0.000000000000001	0.739085133215160	0.000000000000001
52	0.000000000000000	0.739085133215160	0.000000000000000
53	0.000000000000000	0.739085133215161	-0.000000000000000
54	0.000000000000000	0.739085133215161	0.000000000000000
55	0.000000000000000	0.739085133215161	-0.000000000000000

これから，確かに誤差が $O(2^{-n})$ で減ってゆくことが見て取れる．
しかし，函数値で見た誤差は単調減少ではないことも観察される．
2 分法は高速な計算が可能なので，実際には $n = 53$ くらいまでは
あっと言う間に計算でき，それで十分な精度が得られる．

この問題に対しては “真の値” は無いが，2 分法の値はまさに区間演算なので，真の値と見てよいであろう．ちなみに Risa/Asir での計算値は
0.739085133215160641655312087673873404013411758900757464965680 6 $\cdots$

Newton 法

昔は高校でもやったことがある.

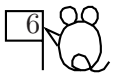


第 n 近似値 x_n から第 $n+1$ 近似値 x_{n+1} を,
点 $(x_n, f(x_n))$ における曲線 $Y = f(X)$ への接線
$Y - f(x_n) = f'(x_n)(X - x_n)$ が x 軸と 交わる点として定める:
$Y = 0$ と置いて

$$-f(x_n) = f'(x_n)(x_{n+1} - x_n) \qquad \therefore \quad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

これを, 二分法と 同じ 方程式で実験してみよう.

FORTRAN プログラム num5-2.f (C 版は num5-2.c)

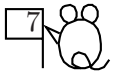


2 分法と同じ方程式を解いている.

```
      PROGRAM NEWTON
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      F(X)=X-DCOS(X)        ! プレゼンの挿絵と同じ形にする
      G(X)=1+DSIN(X)        ! ここでは F'(X) は人間が計算した
      X=1.57D0              ! 近似解の初期値
      EPSLON=1.0D-15
      DO I=1,100            ! 反復回数は適当に大きめに設定
         DX=F(X)/G(X)
         X=X-DX
         WRITE(*,200)I,'-th Iteration: ',X,'; Error: ',DX
         IF (DABS(DX).LT.EPSLON) GO TO 100  !.LT. は不等号<のこと
      END DO
  100 STOP       ! GO TO 文の飛び先は実行文に限る, END 等はだめ
  200 FORMAT(1H ,I2,A15,F18.15,A9,F18.15)
      END
```

- 汎用化するには, 全体を Newton 法の収束値を返す函数に変える.
  更に F, G を外部函数とし, EXTERNAL 宣言するとよい.
- GO TO 100 は直接 STOP (函数にした場合は RETURN) と書いてもよい.
- EPSLON はミスプリではない. 6 文字に納めるため詰めてある.

実行結果

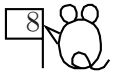


反復回数	近似解	一つ前との差分
1	0.785398038969214	0.784601961030786
2	0.739536131151519	0.045861907817696
3	0.739085178105540	0.000450953045979
4	0.739085133215161	0.000000044890379
5	0.739085133215161	0.000000000000000

Newton 法の収束はおそろしく速い.

今までの近似公式はいずれも1 次の収束.
i.e. 回数に比例して正しい桁数が増えて行く.
これに対し, Newton 法は2 次の収束をする.
i.e. 正しい桁数が常に一つ前の桁数の倍に増える.
初期誤差から見ると, 誤差は反復回数につき2 重指数的に減少: $O(e^{-c\,2^n})$
i.e. 正しい桁数が反復回数について指数的に増加する.

これは級数で例えると $\displaystyle\sum_{n=1}^{\infty}\frac{1}{e^{2^n}}$ の収束の速さと同等で
e^x や $\sin x$ の Taylor 展開に対する $O(e^{-c\,n\log n})$ よりも更に速い.



真の解を a とする．i.e. $f(a)=0$.
話を決めるため，$x_n > a$ として論ずる．
漸化式 $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$ の両辺から a を引くと

$$x_{n+1} - a = x_n - a - \frac{f(x_n)}{f'(x_n)} = -\frac{f(x_n) - (x_n - a)f'(x_n)}{f'(x_n)}$$

ここで，分子は，$f(a)=0$ に注意し平均値の定理を繰り返し用いると

$$\begin{aligned} f(x_n) - (x_n - a)f'(x_n) &= \{f(x_n) - f(a)\} - (x_n - a)f'(x_n) \\ &= (x_n - a)f'(\xi_n) - (x_n - a)f'(x_n) \\ &= (x_n - a)\{f'(\xi_n) - f'(x_n)\} \\ &= (x_n - a)(\xi_n - x_n)f''(\eta_n) \end{aligned}$$

となる．ここに $a < \xi_n < \eta_n < x_n$.
この量は $|f''(x)| \le M_2$ とすれば，$\le M_2|x_n - a|^2$
よって，$|f'(x)| \ge m_1$ と仮定すれば，

$$|x_{n+1} - a| \le \frac{M_2}{m_1}|x_n - a|^2$$

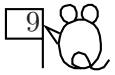
つまり，誤差が一つ前の2 乗で小さくなる2 次の収束をしている．
これは，初期誤差でいうと，

$$\begin{aligned} &\le \cdots \le \Big(\frac{M_2}{m_1}\Big)^{1+2+4+\cdots+2^{n-1}} |x_1 - a|^{2^n} = \Big(\frac{M_2}{m_1}\Big)^{2^n-1}|x_1 - a|^{2^n} \\ &\le C\Big(\frac{M_2}{m_1}|x_1 - a|\Big)^{2^n} = Ce^{-\lambda 2^n} \quad (\lambda = -\log\big(\frac{M_2}{m_1}|x_1 - a|\big)) \end{aligned}$$

つまり，$O\big(\dfrac{1}{e^{\lambda 2^n}}\big)$ の形で小さくなる．

初期値を $\dfrac{M_2}{m_1}|x_1 - a| < 1$ に取らないと収束は保証されない．

応用：函数の作成 `num5-3.f`



これだけ収束が速いと，函数ライブラリの作成に使える．

例：sqrt(x) の最も速い実装
$y^2 - x = 0$ を Newton 法で y につき解くと，漸化式は

$$x_{n+1} = x_n - \frac{x_n^2 - x}{2x_n} = \frac{x_n}{2} + \frac{x}{2x_n}$$

これだけでも十分速い (倍精度の算出に3 回程度の反復で済む．)

更なる工夫：一番時間がかかる割り算を避けて，掛け算だけでやる．
そのため, まず $\frac{1}{y^2} - x = 0$ を Newton 法で解く：

$$x_{n+1} = x_n - \frac{1/x_n^2 - x}{-2/x_n^3} = x_n + \frac{x_n}{2} - \frac{x_n^3 x}{2} = x_n\left(\frac{3}{2} - \frac{x_n^2 x}{2}\right)$$

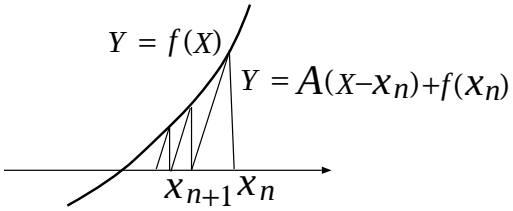
得られた $\frac{1}{\sqrt{x}}$ に x を掛けて $\sqrt{x}$ を求める．
この工夫による計算時間の改良は，函数を1 回使っただけでは分からない．
しかし，数値積分などで何万回も呼ぶと差が出て来る．
cf. 2GHz の CPU は，1 クロックでできる演算を1 秒間に 2G 回実行．
　主な浮動小数演算のクロック数 (Pentium II の場合)：
　　load, store (1)，　　加減乗算 (3)，　　除算 (19)
最近の CPU はパイプライン方式なので, 乗除算も1 クロックのように
書かれているが, パイプが詰まったときは上の比に近くなってしまうので,
差が無くなった訳ではない.

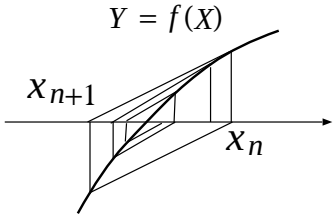
線型反復法　num5-4.f

Newton 法はすばらしいが，毎回 $f'(x)$ を計算する必要がある．
$f'(x)$ の計算は簡単ではないかもしれない．そこで
$f'(x_n)$ を使う代わりに，一定の傾き A (例えば $A = f'(x_1)$) で代用すると

線型反復公式　$x_{n+1} = x_n - \dfrac{f(x_n)}{A}$

この反復法の収束は，初期値のみならず，傾き A の取り方にも依存する．





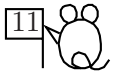
$$|x_{n+1} - a| = \left| x_n - a - \frac{f(x_n) - f(a)}{A} \right| \le \frac{|A - f'(\xi_n)|}{|A|} |x_n - a|$$

だから，A が求める解の近傍で $f'(x)$ の良い近似になっていれば,

$$\frac{|A - f'(\xi_n)|}{|A|} \le \lambda < 1, \quad \text{従って} \quad |x_{n+1} - a| \le \lambda^n |x_1 - a|$$

と，1 次の収束が保証される．
$\cos x = x$ で実行してみると，$A = f'(x_1)$ で2 分法より速い．
しかし，一般には A の選び方で2 分法より遅くなることもある．

Richardson 加速法



$$a_n = a + c_1\lambda_1^n + c_2\lambda_2^n + c_3\lambda_3^n + \cdots,$$
$$(1 > \lambda_1 > \lambda_2 > \lambda_3 > \cdots)$$

という数列の極限値 a の近似値は，大きな n に対して a_n を直接計算するより

$$a_{n+1} = a + c_1\lambda_1^{n+1} + c_2\lambda_2^{n+1} + c_3\lambda_3^{n+1} + \cdots,$$

と上との差をとって

$$t_{1,n} = \frac{a_{n+1} - \lambda_1 a_n}{1-\lambda_1} = a + c_2\frac{\lambda_2-\lambda_1}{1-\lambda_1}\lambda_2^n + c_3\frac{\lambda_3-\lambda_1}{1-\lambda_1}\lambda_3^n + \cdots$$

の形にしておけば，ずっと小さな n に対して a のより良い近似値が
得られるであろう．更に，

$$t_{1,n+1} = \frac{a_{n+2} - \lambda_1 a_{n+1}}{1-\lambda_1} = a + c_2\frac{\lambda_2-\lambda_1}{1-\lambda_1}\lambda_2^{n+1} + c_3\frac{\lambda_3-\lambda_1}{1-\lambda_1}\lambda_3^{n+1} + \cdots$$

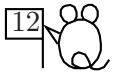
との差を取って，

$$t_{2,n} = \frac{t^1_{n+1} - \lambda_2 t^1_n}{1-\lambda_2} = a + c_3\frac{(\lambda_3-\lambda_1)(\lambda_3-\lambda_2)}{(1-\lambda_1)(1-\lambda_2)}\lambda_3^n + \cdots$$

とすれば，もっとよい近似になるであろう．
この操作は，上の漸近展開が有効な限り 続けることができる．

この計算には，予め $\lambda_1, \lambda_2, \lambda_3, \cdots$ が分かっている必要がある．

近似式の加速への応用　num5-5.f



$f(h)$ が $O(h)$ の近似式のとき，多くの場合に

$$f(h) = c_0 + c_1 h + c_2 h^2 + \cdots$$

という漸近展開を持つ．　(ここに $c_0 = f(0)$ は求めたい真の極限値)

漸近展開とは，$\forall N$ に対し，

$$f(h) - \{c_0 + c_1 h + c_2 h^2 + \cdots + c_N h^N\} = O(h^{N+!})$$

となっていること．(級数は収束しなくてもよい．)

このとき，$h = h_0, h_0/2, h_0/2^2, \ldots, h_0/2^N$ に対して $f(h)$ を計算して得られる数列 f_n は，Richardson 加速が適用できる典型的な例となる：

$$f(\frac{h_0}{2^n}) = c_0 + \frac{c_1 h_0}{2^n} + \frac{c_2 h_0^2}{4^n} + \frac{c_3 h_0^3}{8^n} + \cdots$$

i.e. $\lambda_1 = \frac{1}{2}$, $\lambda_2 = \frac{1}{4}$, $\lambda_3 = \frac{1}{8}$,... で Richardson 加速の用件を満たす．

従って，適当に係数を掛けて1 次結合をとると，$c_1, c_2, \ldots, c_N$ を消去でき，

$$\text{既知の値 (計算値の1 次結合)} = f(0) + O(h_0^{N+1})$$

という式が得られるであろう．

i.e. 1 次の近似式を元に，$N+1$ 次の近似式が作れるであろう．

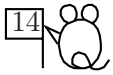
num5-5.f は，前進差分を加速した例．

実際のアルゴリズム　以下，$f_n = f(h_0/2^{n-1})$ と置く

- f_1 を計算する．
- f_2 を計算する．
- $t_{1,1} = \dfrac{f_2 - f_1/2}{1 - 1/2} = \dfrac{2f_2 - f_1}{2 - 1}$ を計算する．
  これが良い近似値なら停止する．
- f_3 を計算する．
- $t_{1,2} = \dfrac{2f_3 - f_2}{2 - 1}$ を計算する．
- $t_{2,1} = \dfrac{2^2 t_{1,2} - t_{1,1}}{2^2 - 1}$ を計算する．これが良い近似値なら停止する．
- f_4 を計算する．
- $t_{1,3} = \dfrac{2f_4 - f_3}{2 - 1}$ を計算する．
- $t_{2,2} = \dfrac{2^2 t_{1,3} - t_{1,2}}{2^2 - 1}$ を計算する．
- $t_{3,1} = \dfrac{2^3 t_{2,2} - t_{2,1}}{2^3 - 1}$ を計算する．これが良い近似値なら停止する．
- ⋮　(停止条件は普通 $|t_{n,1} - t_{n-1,2}| < \varepsilon$ とする．)

$$
\begin{array}{cccc}
f_1 & & & \\
t_{1,1} & f_2 & & \\
t_{2,1} & t_{1,2} & f_3 & \\
t_{3,1} & t_{2,2} & t_{1,3} & f_4 \\
\vdots & & &
\end{array}
$$

Romberg 積分法



元になった近似式が2 次の場合:

$$f(h) = c_0 + c_1 h^2 + c_2 h^4 + \cdots$$

という漸近展開を持つとすれば，$h = h_0, h_0/2, h_0/2^2, \ldots, h_0/2^N$ に対して

$$f(\frac{h_0}{2^n}) = c_0 + \frac{c_1 h_0^2}{4^n} + \frac{c_2 h_0^4}{16^n} + \frac{c_3 h_0^6}{64^n} + \cdots$$

i.e. $\lambda_1 = \frac{1}{4}$, $\lambda_2 = \frac{1}{4^2}$, $\lambda_3 = \frac{1}{4^3}$,... で Richardson 加速の用件を満たす.

従って，2 次の近似式から更に効率良く $2N+2$ 次の近似式が
一般の Richardson 加速と同程度の手間で作れる:

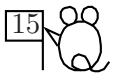
$$\text{既知の値 (計算値の1 次結合)} = f(0) + O(h_0^{2N+2})$$

適用例: 台形公式にこの Richardson 加速法を適用したものが
Romberg 積分法である.

実際の計算アルゴリズムは，1 次の近似式のときの 2^n を 4^n に変えるだけ.

台形公式が上のような h に関する漸近展開を持つことは
先の誤差計算において f の Taylor 展開をもっと高次まで
やっておけば初等的に示せる.
(もちろん, 被積分館数は必要なだけ微分可能でなければならない.)

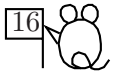
実際のアルゴリズム−2　以下，$f_n = f(h_0/2^{n-1})$ と置く



- f_1 を計算する．
- f_2 を計算する．
- $t_{1,1} = \dfrac{f_2 - f_1/4}{1 - 1/4} = \dfrac{4f_2 - f_1}{4 - 1}$ を計算する．
  これが良い近似値なら停止する．
- f_3 を計算する．
- $t_{1,2} = \dfrac{4f_3 - f_2}{4 - 1}$ を計算する．
- $t_{2,1} = \dfrac{4^2 t_{1,2} - t_{1,1}}{4^2 - 1}$ を計算する．これが良い近似値なら停止する．
- f_4 を計算する．
- $t_{1,3} = \dfrac{4f_4 - f_3}{4 - 1}$ を計算する．
- $t_{2,2} = \dfrac{4^2 t_{1,3} - t_{1,2}}{4^2 - 1}$ を計算する．
- $t_{3,1} = \dfrac{4^3 t_{2,2} - t_{2,1}}{4^3 - 1}$ を計算する．これが良い近似値なら停止する．
- $\vdots$

f_1			
$t_{1,1}$	f_2		
$t_{2,1}$	$t_{1,2}$	f_3	
$t_{3,1}$	$t_{2,2}$	$t_{1,3}$	f_4
$\vdots$			

本日の講義内容の自習課題



1 2 分法のプログラム num5-1.f をコンパイルし, 実行してみて,
講義で述べた収束の様子を確認する.

2 Newton 法のプログラム num5-2.f をコンパイルし, 実行してみて,
講義で述べた収束の様子を確認する.

3 Newton 法を用いて二つの方法で自作した sqrt 函数のスピードを比較する.
num5-3.f をコンパイルし, 実行してみて, 講義で述べたことを確認する.

4 線型反復法の例 num5-4.f をコンパイルし, 実行してみて,
講義で述べた収束の様子を確認する.

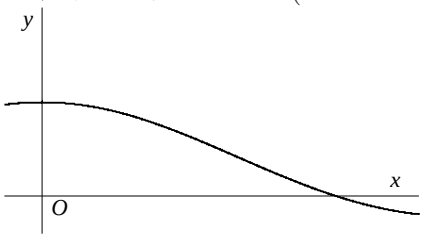
5 前進差分に Richardson 加速を適用したプログラム num5-5.f を
コンパイルし, 実行してみて, その威力を確認する.

本日の範囲の試験予想問題

課題 5.1 2 分法と Newton 法，および線型反復法の原理を説明し，それぞれの長所，短所を述べよ．

問題 5.2 (1) 超越方程式 $e^x = 2x + 1$ は $x = 0$ 以外に解を持つことを示せ．
(2) 上記の解を計算機で近似計算する方法を二つ示せ．

問題 5.3 関数 $y = \dfrac{\sin x}{x}$ は $0 \leq x \leq \pi$ で単調減少し，区間 $[0, 1]$ の値を 1 度ずつ取る (下図参照)．定義域を区間 $[0, 1]$ とするこの関数の逆関数を実装する方法を示せ．(プログラムまで書く必要は無い.)



問題 5.4 上の逆関数を普通に実装すると，$x = 1$ の近くで有効桁数が半分程度に落ちてしまう．この理由を説明し，対策法を述べよ．

問題 5.5 中心差分による 1 階微分の近似式を Richardson 加速する方法を説明せよ．

問題 5.6 函数 $y = f(x)$ は 2 重零点を持つとする．i.e.

$$f(a) = f'(a) = 0,\ f''(a) \neq 0.$$

このとき a を求めるための Newton 法の収束の速さを調べよ．