情報処理 2 第 7 回

十進BASIC (5) 方程式の数値解法

桂田 祐史 (かつらだ まさし)

2011 年 6 月 22 日

この授業用の WWW ページは http://www.math.meiji.ac.jp/~mk/syori2-2011/

1 連絡事項

今日は定番の (理工系の学科のコンピューターの授業では、どこかで必ずやるという) 話題です。かえって数学色が濃い？だれませんように。

2 課題 5B 解説

(授業運用上の理由で、しばらく読めないようにします。)

3 方程式の数値解法 — 反復法による方程式の解法

3.1 イントロ

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びます[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」ます。

ここでは**二分法** (**the bisection method**) と**ニュートン法** (**Newton's method**, **the Newton-Raphson method**) を取り上げます。これらは解析学の学習とも深い関係があります。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」ですが、この証明は解を計算する方法を与えているという意味で、「構成的な」証明であると言えるでしょう) そのものであると考えられます。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出来ますし、実際に陰関数・逆関数の計算に利用することも出来ます。

[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。なお、有限次元の線型方程式 (いわゆる未知数有限個の連立 1 次方程式) が簡単と言っているわけではない。

[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。

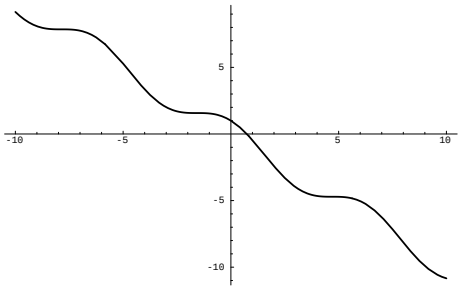
3.2 例題の説明

次の例題から始めます。

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。

この方程式は、一見シンプルですが、式の変形で解を求めることは簡単ではありません (多分…不可能でしょう)。

一方、解が存在することを示すのは比較的簡単です。実際 $f(x) := \cos x - x$ とおいて、f を少し調べてみれば、この方程式にはただ 1 つの実数解があって、それは区間 $(0,1)$ にあることが分かります[3]。f のグラフの概形は、大雑把に言うと、$y = -x$ をサイン・カーブ風に波打たせたものになっています。



コンピューターで描いたグラフを見れば、$f(x) = 0$ が唯一の実数解を持つことは一目瞭然です。この唯一の実数解を求めることを目標にします。

3.3 コンピューターで「解く」とは

コンピューターで方程式を扱う場合、コンピューターならではのやり方があります。有限回の計算で真の解 (無限精度の解, 「厳密解」 (exact solution) と呼ぶ場合が多い) を求めることはあきらめて、真の解を求めるには無限回の演算が必要になるけれど、有限の要求精度を持つ「解」(近似解というべきでしょう) はそこそこの回数の基本的な演算 (四則演算) で求まるような方法を採用する、というものです。従って、アルゴリズムは大抵の場合、繰り返しのあるものになります。

コンピューターによる方程式の数値解法は「繰り返し」が鍵

1. 線型方程式 (1 次方程式) でも、未知関数の個数 N が非常に大きい問題の解法には、反復計算が使われます。
   例えば $N = 10^8$ (原子炉圧力容器の構造計算とか[4]) のとき、消去法計算は無理です。

2. 非線型方程式の場合、問題の自由度が低くても反復計算になる場合がほとんどです。

[3] まず $f'(x) = -\sin x - 1 \leq 0$ $(x \in \mathbf{R})$ 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、中間値の定理によって、方程式 $f(x) = 0$ は区間 $(0,1)$ 内に少なくとも 1 つの解を持つが、f の単調性からそれは $\mathbf{R}$ 全体でただ 1 つの解であることが分かる。■

[4] 本来は、偏微分方程式で、未知数の個数は無限というべき問題だが、適当な離散化により、有限次元の問題になる。

(a) 有限回の四則では exact (厳密) に解けない問題がほとんど。

(b) 「反復法」で多くの問題が解ける。
すなわち、真の解 x^* をある列 $\{x_n\}$ の極限としてとらえ ($\lim_{n\to\infty} x_n = x^*$)、十分大きな n に対して x_n を x^* の近似として採用する。近似解ではあるが、コンピューターで数値計算する限り、有限精度であることは避けられないので、exact な方法 (もしそれがあったとして) とほとんど差がない[5]。

例 3.1 $x^3 - 4x^2 + 3x + 4 = 0$ の実数解を求めよ、という問題の解は、3 次方程式の解に関する **Cardano の公式**から、

$$x = \frac{1}{3}\left(4 - \frac{7}{\sqrt[3]{44 - 3\sqrt{177}}} - \sqrt[3]{44 - 3\sqrt{177}}\right)$$

と求まります。実際的必要から数値を求めたい場合は、$\sqrt{\ }$, $\sqrt[3]{\ }$ を数値計算する必要が生じます。最初から二分法や Newton 法で直接解を計算する方が簡単である可能性が高いです。■

注意 3.2 ここで紹介するのは近似計算であり、近似計算は数学的に厳密ではないから「意味がない」と素朴に考える人がいるかもしれません。これについては以下の二つのことを指摘しておきます。

1. 近似計算ではあっても「状況証拠」くらいにはなって、本当がどうなっているのかを想像することは出来て、役立つことが多い[6]。

2. コンピューターの計算結果に厳密で具体的な精度の保証をつけることができ、例えば解の存在なども証明できる場合がある (**精度保証付き数値計算**)。■

3.4 二分法

計算の原理は、**中間値の定理**「連続な関数 $f\colon [a,b] \to \mathbf{R}$ について、$f(a)$ と $f(b)$ の符号が異なれば、(a,b) に解が少なくとも 1 つ存在する」と、その**区間縮小法**による証明に基づきます。

$[a,b]$ に解が存在するならば、2 つに分割した区間 $[a,(a+b)/2]$, $[(a+b)/2,b]$ のどちらかに存在します (両方に存在することもある) が、どちらにあるか判断できれば、繰り返すことで区間の幅を半分半分にして行けて、解を追い詰めることが出来る、ということです。(詳しいことは次の小節で — 暇な時に読んでね。)

以下にサンプル・プログラムを示します。少し長いですが、心臓部分 (プログラム後半部分) は (区間縮小法を理解していれば) 難しくないでしょう。

---

[5]実際、連立 1 次方程式の解を求めるために、**Gauss の消去法**などの exact な計算法が知られているが、**CG 法** (共役勾配法) などの反復法が採用されることも多い。

[6]ここで引き合いに出すのは、少々こじつけかもしれないが、アルキメデス (BC 287–212, シラクサに生まれ、シラクサに没する) の『方法』に書いてあるという言葉「ある種の問題は、まず工学的な方法で答が明らかになってしまう。もちろん後で幾何学的に証明を付けなくてはいけないのだけれども、それでも最初から答がわかっているのと、一から考えなくてはならないのとでは雲泥の差がある」— 木村俊一, 『天才数学者はこう解いた、こう生きた』、講談社から引用。つまり 2000 年の歴史のある言い訳。

bisection.BAS

```
REM bisection.BAS ---      ^^ca^^ac^^cb^^a1^^ca^^b6     ^^d6^^bd^^cc^^be^^cb^^a1^^cb^^a4
   f(x)=0 う董@押
REM     ^^d5^^a1 1000 掘^^e2^^a1^^bc^^c9^^a4^^cb^^a4 靴^^c6^^a4   ^^c4^^b6^^db^^b4^^d8^^bf
    17 掘
REM ^^c4^^b6^^db^^b4^^d8^^bf            ^^c8^^a4        ^^ca^^a4       韻          1000 掘
^^e2^^a1^^bc^^c9^^a4^^c7^^b9  精  ^^d9^^a4 ζ  ☒      世
REM OPTION ARITHMETIC DECIMAL_HIGH
OPTION ARITHMETIC NATIVE
REM ^^f2^^a4^^ad^^a4 燭ぁ◎  ☒ f(x)=0 ぁ    ～
FUNCTION F(x)
   LET F=COS(X)-X
END FUNCTION
REM ------------------------------------------------
LET FMT$="---%.############### "
LET FMT2$=FMT$&FMT$
INPUT PROMPT "左^^c3^^bc、右^^c3^^bc=": A,B
LET EPS=(B-A)*1.0e-14
REM -------------   ☒^^cf^^a4 靴拭^^cd^^a4 ϒ船Д^^c3^^a5 ------------------
LET FA=F(A)
LET FB=F(B)
IF FA=0 THEN
   PRINT A;"^^cf^^b2  ^^c7^^a4 后"
   STOP
ELSEIF FB=0 THEN
   PRINT B;"^^cf^^b2  ^^c7^^a4 后"
   STOP
ELSEIF (FA > 0 AND FB > 0) OR (FA < 0 AND FB < 0) THEN
   PRINT "f(a),f(b)ぁ    ^^e6^^a4^^ac^^c6^^b1 じ^^c7^^a4"
   STOP
END IF
REM ------------      ^^ca^^ac^^cb^^a1^^ca^^b6     ^^d6^^bd^^cc^^be^^cb^^a1^^cb^^a4
^^c2^^b9   -------------
LET MAXITR=100
FOR i=1 TO MAXITR
   LET C=(A+B)/2
   LET FC=F(C)
   IF FC=0 THEN
      PRINT "^^f2^^a4^^ac^^b8 ☒^^c4^^a4 ☒  ^^de^^a4 靴拭"
      PRINT USING FMT$: C
      STOP
   ELSEIF (FA>0 AND FC<0) OR (FA<0 AND FC>0) THEN
      REM 左^^c2^^a6[A,C]^^cb^^b2^^f2^^a4^^ac^^a4 △
      LET B=C
      LET FB=FC
   ELSE
      REM 右^^c2^^a6[C,B]^^cb^^b2^^f2^^a4^^ac^^a4 △
      LET A=C
      LET FA=FC
   END IF
   PRINT USING "###": I;
   PRINT USING FMT2$: A,B;
   PRINT FA;FB
   IF B-A < EPS THEN
      PRINT "供^^d6^^a4    ☒☒^^ca^^ac 小さく^^ca^^a4  ^^de^^a4 靴拭☒    ＼ぁ^^c9^^bd 示し
^^de^^a4 后"
      PRINT USING FMT$: (A+B)/2
      STOP
   END IF
NEXT I
PRINT "供^^d6^^a4    ☒^^cf^^bd^^ca^^ac 小さく^^ca^^a4  ^^de^^a4 擦  ^^c7^^a4 靴拭"
END
```

実行すると「区間の左端、右端」を尋ねてくる。例えば 0,1 と答える。

bisection.TXT

```
左^^c3^^bc、右^^c3^^bc=0,1
  1   0.500000000000000    1.000000000000000  .377582561890373 -.45969769413186
  2   0.500000000000000    0.750000000000000  .377582561890373 -1.83111311261791E-2
  3   0.625000000000000    0.750000000000000  .185963119505218 -1.83111311261791E-2
  4   0.687500000000000    0.750000000000000  8.53349461524715E-2 -1.83111311261791E-2
  5   0.718750000000000    0.750000000000000  3.38793724180665E-2 -1.83111311261791E-2
  6   0.734375000000000    0.750000000000000  7.87472545850132E-3 -1.83111311261791E-2
  7   0.734375000000000    0.742187500000000  7.87472545850132E-3 -5.19571174375921E-3
  8   0.738281250000000    0.742187500000000  1.34514975180511E-3 -5.19571174375921E-3
(    ^^ce^^ac)
 42   0.739085133214985    0.739085133215212  2.93876034618279E-13 -8.65973959207622E-
 43   0.739085133215099    0.739085133215212  1.03583808197527E-13 -8.65973959207622E-
 44   0.739085133215156    0.739085133215212  8.54871728961371E-15 -8.65973959207622E-
 45   0.739085133215156    0.739085133215184  8.54871728961371E-15 -3.90798504668055E-
 46   0.739085133215156    0.739085133215170  8.54871728961371E-15 -1.53210777398272E-
 47   0.739085133215156    0.739085133215163  8.54871728961371E-15 -3.44169137633799E-
供^^d6^^a4    ☒☒^^ca^^ac 小さく^^ca^^a4  ^^de^^a4 靴拭☒    ＼ぁ^^c9^^bd 示
し^^de^^a4 后
   0.739085133215159
```

なお、この計算では要求精度 (区間の幅がどこまで小さくなったら反復を停止するか) を 1e-14 (意味は 1×10^{-14} という意味) としてありますが、これは演習に用いている十進BASICの通常の演算精度が、10 進法 15 桁であることから決めたものです。

やってみよう　$x^2-2=0$ を解くことで、$\sqrt{2}$ を求めてみよ (これは課題 8B の一部である)。

3.5　参考: 二分法 (bisection method) の原理

(情報処理教室で実習中にここを読む時間はほとんどないでしょうから、余裕のあるときに読んで下さい。)

微積分で基本的な**中間値の定理**を復習しましょう。

定理 3.3 (中間値の定理) $f:[\alpha,\beta]\to\mathbf{R}$ は連続関数で、$f(\alpha)$ と $f(\beta)<0$ の符号が異なるとき、$f(c)=0$ 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りますが、代表的なものに**区間縮小法**を使ったものがあります。それは以下のような筋書きです。

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 $f(x) = 0$ の近似解を求める次の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す)。

二分法のアルゴリズム

(1) 要求する精度 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 (3) に戻る。そうでなければ $\dfrac{a+b}{2}$ を解として出力する。
($[a, b]$ 内に真の解が存在することが分かるので、a と b を出力することにも意味がある。)

注意 3.4 (反復法の停止則 (初めて見るときはパスして構いません)) このような反復法では、繰り返しをどこで止めるかという停止則が重要です。方程式の数値解法の場合、

近似解 x_n における f の値が、丸め誤差を考慮して、0 と区別がつかなくなったら停止する

というのが「まあまあ筋の通った」停止則とされています[7]。しかし、これを実際に遂行するのはあまり簡単ではないので、世の中に出回っているテキストでは、次のどちらかでお茶を濁していることが多いです。

(1) ほどほどに小さい数 ε を (かなりいい加減に) 取り、$\|x_n - x_{n+1}\| \le \varepsilon$ となったら停止する。

[7]杉原正顯、室田一雄、『数値計算法の数理』、岩波書店などを見よ。

(2) ほどほどに小さい数 ε を (かなりいい加減に) 取り、$|f(x_n)| \leq \varepsilon$ となったら停止する。

上の例の計算で採用した停止則も、実はかなりいいかげんです (方針 (1) に近いです)。二分法の場合、$f(x)$ の計算結果の符号が正しく計算できている間は、真の解は確実に区間の中に含まれていることになりますが、実際に x が解の近くになると、$|f(x)|$ は 0 に近くなり、丸め誤差よりも小さくなってしまうと、$f(x)$ の計算値は全然信用できないものになります。 ■

3.6 Newton 法

論より run で行ってみましょう[8]。

[8]BASIC では、プログラムを動かすことを「run させる (走らせる)」と言うことがあります。

newton.BAS

```
REM newton.BAS --- Newton^^cb^^a1あ f(x)=0 う董@押
REM 新  x=x+Δ  x, Δ  x=-f(x)/f'(x) ^^c8^^a4  い α 臆 充 阿^^c7^^b6    ☒
^^f2^^b9^^b9^^bf
REM     ^^d5^^a1 1000   掘^^e2^^a1^^bc^^c9^^a4^^cb^^a4   靴^^c6^^a4
^^c4^^b6^^db^^b4^^d8^^bf    17 掘
REM ^^c4^^b6^^db^^b4^^d8^^bf       ^^c8^^a4    ^^ca^^a4  韻      1000 掘
^^e2^^a1^^bc^^c9^^a4^^c7^^b9   精  ^^d9^^a4 ζ  ☒^^cd^^a4 ☒

REM そう痢    PRINT USING う論  ☆あ^^c5^^ac  ☒  ^^c4^^be す
REM OPTION ARITHMETIC DECIMAL_HIGH
OPTION ARITHMETIC NATIVE
REM -----------------------------------------------------------------
REM ^^f2^^a4^^ad^^a4 燭ぁ◎  ☒ f(x)=0 う文    f(x) ぁ    〜
FUNCTION F(x)
   LET F=COS(X)-X
   REM LET F=x^2-2
END FUNCTION
REM f'(x)ぁ    〜
FUNCTION dfdx(x)
   LET DFDX=-SIN(x)-1
   REM dfdx=2*x
END FUNCTION
REM -----------------------------------------------------------------
INPUT PROMPT "宗●  ^^cd^^a1◆^^d7^^b5  精    (1e-14^^ca^^a4  )=": X,EPS
REM Newton^^cb^^a1ぁ^^c2^^b9
LET MAXITR=100
FOR i=1 TO MAXITR
   LET dx=-f(x)/dfdx(x)
   LET x=x+dx
   PRINT USING "f(--%.################)=---%.##^^^^": x, f(x)
   IF ABS(dx)<EPS THEN
      PRINT USING "Δ x=---%.##^^^^":dx
      STOP
   END IF
NEXT I
PRINT "修正    |Δ x| ^^cf^^bd^^ca^^ac 小さく^^ca^^a4  ^^de^^a4 擦  ^^c7^^a4
靴拭"
END
```

実行すると、「初期値と要求精度」を尋ねて来るので、例えば 1,1e-14 と答えます。

newton.TXT

```
宗●  ^^cd^^a1◆^^d7^^b5  精    (1e-14^^ca^^a4  )=1,1e-14
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
Δ x=   0.00E+00
```

3.7 参考: Newton 法の原理

(情報処理教室で実習中にここを読む時間はほとんどないでしょうから、余裕のあるときに読んで下さい。)

Newton 法は非線形方程式を解くための代表的な方法です。

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 が得られているとき、線形化写像

$$x \mapsto f(x_0) + f'(x_0)(x - x_0)$$

の零点 $x = x_0 - [f'(x_0)]^{-1} f(x_0)$ は、x_0 よりも良い近似解になっているだろう (実際に適当な条件下でこれは正当化できます)、という考えから導かれるものです。

すなわち、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[9]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:

$$f(x_*) = 0$$

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ります。

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。

[9]Newton 法が収束するための十分条件は色々知られているが、ここでは説明しません。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ありますが、山本哲朗『数値解析入門』サイエンス社 (2003) がお勧め。

3.8 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でしょうか？それぞれの長所・短所をあげて比較してみます。

3.8.1 二分法の特徴

(i) 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。

(ii) 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)。

(iii) $f(\alpha)$ と $f(\beta)$ の符号が異なる α と β が見つかっていれば、確実に近似解が求まる。

(iv) 収束は遅い。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。
(例えば 10 進 1000 桁 (2 進 3000 桁以上) 求める場合を考えてみよう。)

3.8.2 Newton 法の特徴

(i) 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。

(ii) 微分可能であっても、f' の実際の計算が難しい場合は適用困難になる。

(iii) 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)。

(iv) 適切な初期値を探すことが難しい場合もある。

(v) 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 1 段進むごとに 2 倍になる)、非常に速い。
(2 次収束モードに入れば、10 進 1000 桁だって楽勝！)

3.8.3 とりあえずの結論

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。」というところだろうか。

3.9 レポート課題 7

プログラムとその実行結果、その説明の 3 点を含んだレポートを TEX を使って執筆し、PDF ファイル `kadai7.pdf` を提出せよ。
二分法、Newton 法のいずれかを用いること (余裕があれば両方で解き比べてみること)。

(1) 与えられた正数 a に対して $\sqrt{a}$ を計算するプログラムを作り、$\sqrt{2}$, $\sqrt{3}$, $\sqrt{5}$ を計算し、組込み関数 `SQR()` の結果と比較せよ。出来れば 1000 桁演算モード (`OPTION ARITHMETIC DECIMAL_HIGH`) でやってみよう。

(2) 与えられた $a \in (0, \infty)$ に対して、(指数関数 EXP() は利用してもよいが、対数関数 LOG() は利用せずに) $\log a$ を計算するプログラムを作り、$\log 10$ を計算し、組込み関数 LOG() の結果と比較せよ。(これは普通の演算モードか、OPTION ARITHMETIC NATIVE でやるのが良いでしょう。)

ヒント

- 紹介したサンプル・プログラム (bisection.BAS, newton.BAS) を書き直すという手順で作成できる。
- $\sqrt{a}$ は例えば $x^2 - a = 0$ の正の解と解釈できる ($x - \dfrac{a}{x} = 0$ の解とする手もあるが…)。
- $\log a$ は $\exp x - a = 0$ の実数解と解釈できる。