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)

2012 年 5 月 30 日 (授業終了後修正版)

この授業用の WWW ページは http://www.math.meiji.ac.jp/~mk/syori2-2012/

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。

1 方程式の数値解法 — 反復法による方程式の解法

1.1 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

1. **二分法** (**the bisection method**)
2. **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** (Cantor's intersection theorem) 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。

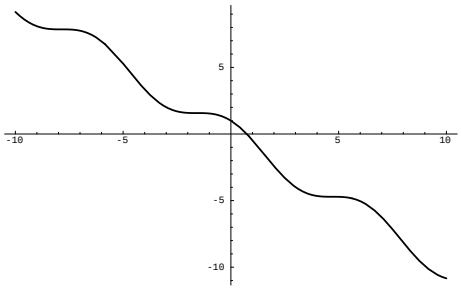
1.2 例題の説明

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。

1.3 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) とも呼ばれる) 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**近似解**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 — **反復法** — 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 1.1 $x^3 - 4x^2 + 3x + 4 = 0$ 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 1.2 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**)。■

厳密解の「厳密」は、論理的に厳密というのとは関係がなくて、論理的に厳密であることと、近似解であることは相反するものではない、という言い方も出来るでしょう。

1.4 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 $\left[a, \frac{a+b}{2}\right], \left[\frac{a+b}{2}, b\right]$ 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。)

[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。

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。

bisection.BAS

```
REM bisection.BAS --- 二分法（区間縮小法）で f(x)=0 の近似解を求める
REM 注意！ 1000 桁モードにしても超越関数は 17 桁
REM 超越関数を使わなければ 1000 桁モードで高精度の近似解を得る
REM 解きたい方程式 f(x)=0 の定義
FUNCTION F(x)
   LET F=COS(X)-X
END FUNCTION
REM -------------------------------------------------------
LET FMT$="---%."&REPEAT$("#",15)&" "
LET FMT2$=FMT$&FMT$
INPUT PROMPT "左端、右端=": A,B
LET EPS=(B-A)*1.0e-14
REM ------------- 入力した値のチェック ------------------
LET FA=F(A)
LET FB=F(B)
IF FA=0 THEN
   PRINT A;"は解です。"
   STOP
ELSEIF FB=0 THEN
   PRINT B;"は解です。"
   STOP
ELSEIF (FA > 0 AND FB > 0) OR (FA < 0 AND FB < 0) THEN
   PRINT "f(a),f(b) の符号が同じです"
   STOP
END IF
REM ------------ 二分法（区間縮小法）を実行 -------------
PRINT USING "  0"&FMT2$: a,b
LET MAXITR=1000
FOR i=1 TO MAXITR
   LET C=(A+B)/2
   LET FC=F(C)
   IF FC=0 THEN
      PRINT "解が見つかりました。"
      PRINT USING FMT$: C
      STOP
   ELSEIF (FA>0 AND FC<0) OR (FA<0 AND FC>0) THEN
      REM 左側 [A,C] に解がある
      LET B=C
      LET FB=FC
   ELSE
      REM 右側 [C,B] に解がある
      LET A=C
      LET FA=FC
   END IF
   PRINT USING "###": I;
   PRINT USING FMT2$: A,B
   REM PRINT FA;FB
   IF B-A < EPS THEN
      PRINT "区間の幅が十分小さくなりました。中点を表示します。"
      PRINT USING FMT$: (A+B)/2
      STOP
   END IF
NEXT I
PRINT "区間の幅は十分小さくなりませんでした。"

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

```
左端、右端=0,1
  0    0.000000000000000     1.000000000000000
  1    0.500000000000000     1.000000000000000
  2    0.500000000000000     0.750000000000000
  3    0.625000000000000     0.750000000000000
  4    0.687500000000000     0.750000000000000
  5    0.718750000000000     0.750000000000000
  6    0.734375000000000     0.750000000000000
  7    0.734375000000000     0.742187500000000
  8    0.738281250000000     0.742187500000000
(中略)
 42    0.739085133214989     0.739085133215216
 43    0.739085133215103     0.739085133215216
 44    0.739085133215160     0.739085133215216
 45    0.739085133215160     0.739085133215188
 46    0.739085133215160     0.739085133215174
 47    0.739085133215160     0.739085133215167
区間の幅が十分小さくなりました。中点を表示します。
   0.739085133215164
```

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 16 桁程度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 7 の一部である)。

1.5　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 1.3 (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 1.4 (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。

(2) ほどほどに小さい数 ε を (かなりいい加減に) 取り、$|f(x_n)| \le \varepsilon$ 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。■

[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。

1.6 Newton 法

論より run で行ってみましょう[8]。

newton.BAS

```
REM newton.BAS --- Newton法で f(x)=0 の近似解を求める
REM 新 x=x+Δ x, Δ x=-f(x)/f'(x) という漸化式で近似解更新
REM 注意！ 1000 桁モードにしても超越関数は 17 桁
REM 超越関数を使わなければ 1000 桁モードで高精度の近似値が求められる
REM その場合は PRINT USING の書式等を適当に直す
REM OPTION ARITHMETIC DECIMAL_HIGH
OPTION ARITHMETIC NATIVE
REM -----------------------------------------------------------------
REM 解きたい方程式 f(x)=0 の左辺 f(x) の定義
FUNCTION F(x)
   LET F=COS(X)-X
   REM LET F=x^2-2
END FUNCTION
REM f'(x) の定義
FUNCTION dfdx(x)
   LET DFDX=-SIN(x)-1
   REM dfdx=2*x
END FUNCTION
REM -----------------------------------------------------------------
INPUT PROMPT "初期値、要求精度 (1e-14 など)=": X,EPS
REM Newton法を実行
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 "修正量 |Δ x| は十分小さくなりませんでした。"
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。

[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.TXT

```
初期値、要求精度(1e-14など)=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
```

1.7 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 法で作った点列が収束するための十分条件は色々知られていますが、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、何といっても、Newton 法の研究で世界的に著名な山本先生による、山本哲朗『数値解析入門』サイエンス社 (2003) をお勧めしておきます。

1.8 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。

1.8.1 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。)

1.8.2 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！)

1.8.3 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。

1.9 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 Oh-o! Meiji レポートシステムを使って提出せよ。締め切りは 6 月 12 日 (火曜) 18:00 とします。

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)。

以下、d のところを、最初は d としていましたが、例題プログラム中の区間 $[a, b]$ の a と名前が衝突して、詰まらない混乱を招くので、変更しました (2012 年 5 月 30 日授業終了後の修正)。

(1) 与えられた正数 d に対して $\sqrt{d}$ 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 $d \in [-1, 1]$ に対して (普通の三角関数は利用してもよいが、逆三角関数は利用せずに) $\sin^{-1} d (= \arcsin d)$ を計算するプログラムを作り、$\sin^{-1}\left(\frac{1}{2}\right)$ を計算し、組込み関数 ASIN() 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。
- $\sqrt{d}$ は例えば $x^2 - d = 0$ の正の解と解釈できる ($x - \frac{d}{x} = 0$ の解とする手もあるが…)。
- $\sin^{-1} d$ は $\sin x - d = 0$ の実数解と解釈できる。

2 レポート課題 5B 解説

レポート課題 5B[10] について簡単に解説します。

「課題 5B への取り組み方について」[11] で書いたように、

(1) 10 個の頂点の座標を計算で求めてしまう,

(2) タートルグラフィックスを使う,

(3) 10 個の頂点の座標をとにかく (描いた図から読み取る方法でも良いから) 求めてしまう,

(4) 正五角形の頂点を一つおきに結んで星形を描き PAINT で許してもらう

などの方法があります。

2.1 頂点の座標を計算で求める方法

2.1.1 まずは正五角形から

まず、正五角形を描くことを考えましょう。与えられた自然数 n に対して、円周を n 等分する点の座標を表す式は、あちこちで習っていますね。一周 2π を n 等分した角は $\frac{2\pi}{n}$ ですから、

$$\theta_k := \frac{2\pi k}{n} \quad (k = 0, 1, \cdots, n-1)$$

[10] http://www.math.meiji.ac.jp/~mk/syori2-2012/jouhousyori2-2012-05/node7.html
[11] http://www.math.meiji.ac.jp/~mk/syori2-2012/jouhousyori2-2012-06/node2.html

として、

$$x_k := r\cos\theta_k, \quad y_k := r\sin\theta_k \quad (k = 0, 1, \cdots, n-1)$$

とおくと、(x_k, y_k) $(k = 0, 1, \ldots, n-1)$ は、原点を中心とする半径 r の円周の n 等分点を与えます。

簡単のため $r = 1$ として、また角度が分かりやすいように (?) 単位を度で表すことにして、点の番号を 0 からでなく 1 から振ることにして、また頂点をずらす角度 $\phi = 90^\circ$ を導入して (真上に頂点が来ると星がきれい)

$$\theta_j := \frac{360^\circ}{n}(j-1) \quad (j = 1, 2, \cdots, n),$$
$$x_j := \cos(\theta_j + \phi), \quad y_j := \sin(\theta_j + \phi) \quad (j = 1, 2, \cdots, n)$$

とおきます (ここらへんは好みの問題で、どうでも良いです)。

次のプログラムで正五角形の輪郭が描けます。

```
REM 正五角形を描く
OPTION ANGLE DEGREES
LET n=5
LET DT=360/n
LET p=90
REM 頂点の座標を求める
DIM x(n),y(n)
FOR j=1 TO n
   t=(j-1)*DT+p
   PRINT t
   LET x(j)=COS(t)
   LET y(j)=SIN(t)
NEXT j
REM 正五角形を描く
SET WINDOW -1,1,-1,1
FOR j=1 TO n
   PLOT LINES : x(j),y(j);
NEXT j
PLOT LINES : x(1),y(1)
END
```

正五角形の内部を塗るのは簡単で、最後 (END 行の前) に次の 2 行を加えるだけです。

```
SET AREA COLOR "red"
MAT PLOT AREA : x,y
```

pentagon.BAS

```
REM pentagon.BAS --- 正五角形を描く
OPTION ANGLE DEGREES
LET n=5
LET DT=360/n
LET p=90
REM 頂点の座標を求める
DIM x(n),y(n)
FOR j=1 TO n
   LET t=(j-1)*DT+p
   PRINT t
   LET x(j)=COS(t)
   LET y(j)=SIN(t)
NEXT j
REM 頂点を順に結んで正五角形を描く
SET WINDOW -1,1,-1,1
FOR j=1 TO n
   PLOT LINES : x(j),y(j);
NEXT j
PLOT LINES : x(1),y(1)
REM 塗る
SET AREA COLOR "red"
MAT PLOT AREA: x,y
END
```

2.1.2 素朴に星形

正五角形の頂点を一つおきにたどると星になります。素朴にやると

```
PLOT LINES : x(1),y(1);x(3),y(3),x(5),y(5);x(2),y(2);x(4),y(4);x(1),y(1)
```

となるでしょうか (これを FOR NEXT を使って書くことも出来ますが、自主的な課題としましょう)。

kadai5b0.BAS

```
REM kadai5b0.BAS --- 線画で星
OPTION ANGLE DEGREES
LET n=5
LET DT=360/n
LET p=90
REM 正五角形の頂点の座標を求める
DIM x(n),y(n)
FOR j=1 TO n
   LET t=(j-1)*DT+p
   PRINT t
   LET x(j)=COS(t)
   LET y(j)=SIN(t)
NEXT j
REM 星を描く（線画）
SET WINDOW -1,1,-1,1
PLOT LINES : x(1),y(1);x(3),y(3);x(5),y(5);x(2),y(2);x(4),y(4);x(1),y(1)
END
```

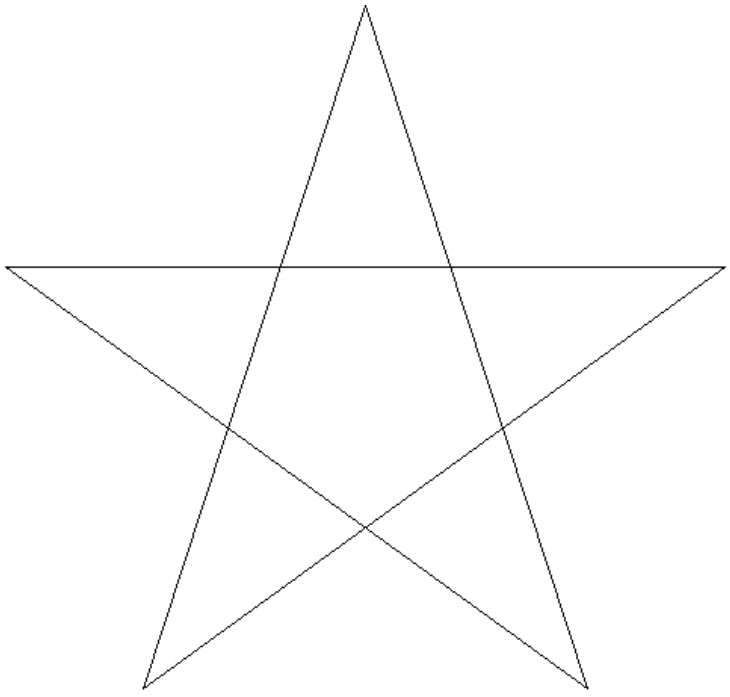


図 1: 線画で星

次のプログラムは、なるべく避けようと言った PAINT を使って正五角形を描くプログラムです。

kadai5b1.BAS

```
REM kadai5b1.BAS --- 星形を描く
OPTION ANGLE DEGREES
LET n=5
LET DT=360/n
LET p=90
REM 正五角形の頂点の座標を求める
DIM x(n),y(n)
FOR j=1 TO n
   LET t=(j-1)*DT+p
   PRINT t
   LET x(j)=COS(t)
   LET y(j)=SIN(t)
NEXT j
SET WINDOW -1,1,-1,1
DRAW grid
SET AREA COLOR "red"
PLOT LINES : x(1),y(1);x(3),y(3);x(5),y(5);x(2),y(2);x(4),y(4);x(1),y(1)

REM 以下どろくさく塗る (PAINT6 連発)
PAINT 0,0
FOR t=1 TO 5
   paint 0.9*x(t),0.9*y(t)
NEXT t
END
```

なお、PLOT LINES を強引に PLOT AREA に置き換えると、図 3 のように真ん中が白く抜けた失敗図が出来ます。

なお、PLOT LINES の前に SET LINE COLOR "red" とすると、星形の中がきれいに塗れます (輪郭線は消えますが)。

2.1.3 頑張って星形の 10 個の頂点を計算する

紙の上に星形を (ある程度正確に) 描いて、頂点の座標を求めましょう。頂点は「外側の円周上にあるもの」と「内側の円周上にあるもの」の二種類あり、それぞれ正五角形の頂点をなすことは容易に分かります。

半径 1 の円に内接する星形とすると、外側の円周上にある頂点 (x_j, y_j) $(j = 1, 2, 3, 4, 5)$ は、上で提示した式で計算出来ることになります。

内側の円周上にある頂点 $(\overline{x}_j, \overline{y}_j)$ $(j = 1, 2, 3, 4, 5)$ は (適当に番号を振ります)、次の性質を持つことは明らかです。

(1) ある正数 r $(0 < r < 1)$ を半径とする原点中心の円周上にある。

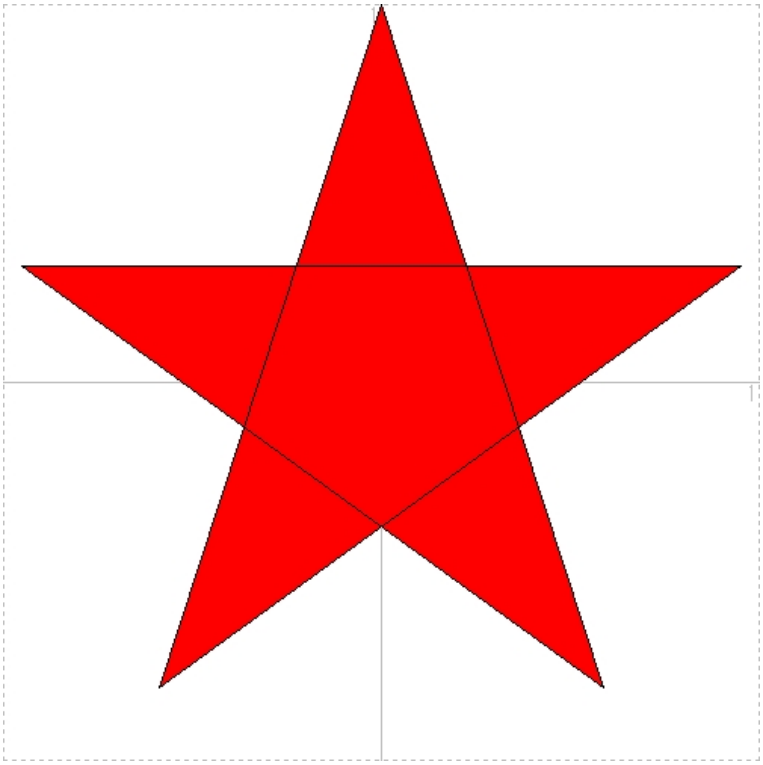


図 2: PAINT 六連発で描きました。

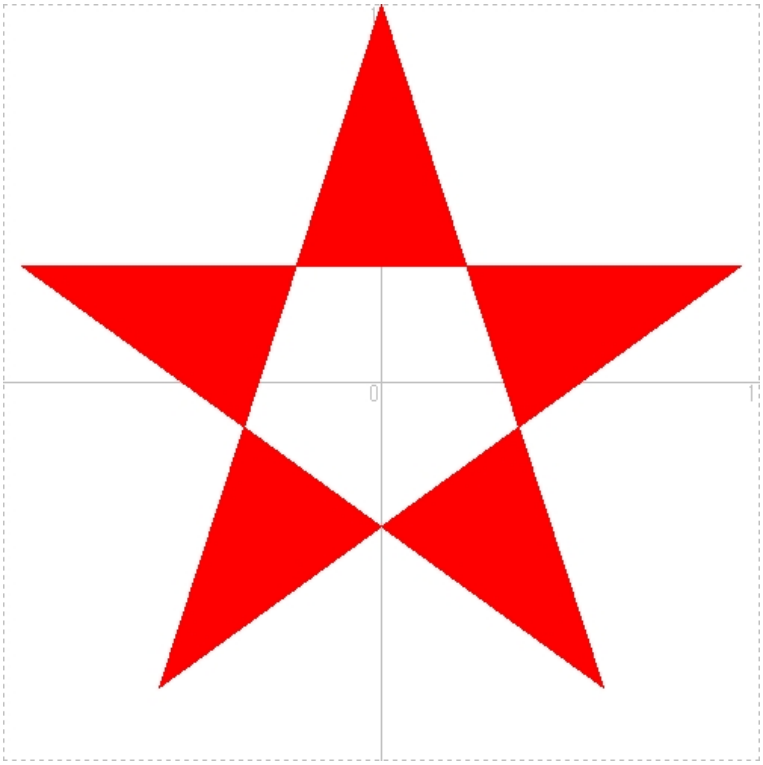


図 3: PLOT AREA: x(1),y(1);x(3),y(3);... とすると

(2) $(\overline{x}_j, \overline{y}_j)$ の角度は、(x_j, y_j) の角度と (x_{j+1}, y_{j+1}) の角度のちょうど真ん中 (ただし $(x_6, y_6) = (x_1, y_1)$ とする)。

r さえ求まれば、後は正五角形を描くのとほとんど同様の計算です。とりあえず r はユーザーに入力してもらうことにすると、次のようなプログラムが書けます。

kadai5b4.BAS

```
REM kadai5b4.BAS --- 星を描く（後一歩バージョン）
OPTION ANGLE DEGREES
SET WINDOW -1,1,-1,1
DIM X(10),Y(10)
INPUT PROMPT "内側の円の半径 r": r
LET DT=360/10
FOR j=1 TO 10
   LET T=(j-1)*DT+90
   REM j が奇数か偶数かで場合分け
   IF MOD(j,2)=1 THEN
      LET x(j)=COS(t)
      LET y(j)=SIN(t)
   ELSE
      LET x(j)=r*COS(t)
      LET y(j)=r*SIN(t)
   END IF
NEXT j
SET AREA COLOR "red"
MAT PLOT AREA : x,y
END
```

r (< 1) の値を色々変えて試してみましょう。$r = 0.4$ くらいが良さそうです。

正確な星形を与える r は、図形の相似比を考えたりして求められます。詳細は自分で解きたい人のお楽しみで取っておきますが (あしからず)、結果は

$$r = \frac{\sin 18^\circ}{\cos 36^\circ} = \frac{\cos 72^\circ}{\cos 36^\circ} = \frac{(\sqrt{5}-1)/4}{(\sqrt{5}+1)/4} = \frac{\sqrt{5}-1}{\sqrt{5}+1} = \frac{3-\sqrt{5}}{2} = 0.38196601\cdots$$

です。

kadai5b4.BAS の

```
INPUT PROMPT "内側の円の半径 r":  r
```

を

```
LET r=(SQR(5)-1)/(SQR(5)+1)
```

に変えると、図 4 のようにきれいな星が描けます。

図 4: 完成版

2.2 タートルグラフィックスで描く方法

星形の輪郭を辿ってみましょう。いつも進む距離は同じで、曲がり方が、右に 144°, 左に 72°, ··· と交互になっている、と分かります。すると、

タートルグラフィックスで星を描く

```
FOR i=1 TO 5
  CALL walk(L)
  CALL right(144)
  CALL walk(L)
  CALL left(72)
NEXT i
```

とすれば星形の輪郭が描けるはずです。

2.2.1 TURTLESTAR1.BAS

```
REM TURTLESTAR1.BAS --- タートルグラフィックスで星の輪郭を描く
OPTION ANGLE DEGREES
REM right(),left(),walk(),jump()
REM 初期化
SUB init
   LET direction=0
```

```
   LET xp=0
   LET yp=0
END SUB
REM 右に曲がる
SUB right(t)
   LET direction=direction-t
END SUB
REM 左に曲がる
SUB left(t)
   LET direction=direction+t
END SUB
REM s 歩ジャンプする
SUB jump(s)
   LET  xp=xp+s*COS(direction)
   LET  yp=yp+s*SIN(direction)
   PLOT LINES: xp,yp
END SUB
REM s 歩進む
SUB walk(s)
   PLOT LINES: xp,yp;
   CALL jump(s)
END SUB
REM ----------- start ------------
LET  L=100
SET WINDOW -2*L,2*L,-2*L,2*L
CALL init
FOR i=1 TO 5
   CALL walk(L)
   CALL right(144)
   CALL walk(L)
   CALL left(72)
NEXT i
END
```

2.2.2 TURTLESTAR3.BAS

上のプログラムに PAINT を一つ加えると塗り潰しできますが、MAT PLOT AREA を使って塗るにはどうしたら良いか、少し工夫をしてみましょう。

亀に記憶力を持たせて、星形の道を辿ってから、通過地点の座標を答えさせるという手もあります。しかし高機能の亀はらしくないので、今どこにいるかを尋ねる imadoko(x,y) というサブルーチンを作って、それを使って描くようにしたのが次のプログラムです。

```
REM TURTLESTAR3.BAS --- タートルグラフィックスで星を描く
```

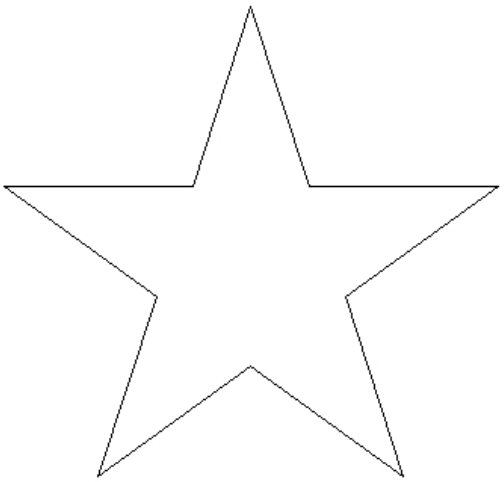


図 5: タートルグラフィックスで星を描く (輪郭線バージョン)

```
OPTION ANGLE DEGREES
REM right(),left(),walk(),jump()
REM 初期化
SUB init
   LET direction=0
   LET xp=0
   LET yp=0
END SUB
REM 右に曲がる
SUB right(t)
   LET direction=direction-t
END SUB
REM 左に曲がる
SUB left(t)
   LET direction=direction+t
END SUB
REM s 歩ジャンプする
SUB jump(s)
   LET  xp=xp+s*COS(direction)
   LET  yp=yp+s*SIN(direction)
   PLOT LINES: xp,yp
END SUB
```

```
REM s 歩進む
SUB walk(s)
   PLOT LINES: xp,yp;
   CALL jump(s)
END SUB
REM 現在位置を答える
SUB imadoko(x,y)
   LET x=xp
   LET y=yp
END SUB
REM ----------- start ------------
LET  L=100
SET WINDOW -2*L,2*L,-2*L,2*L
CALL init
DIM x(10),y(10)
LET n=1
FOR i=1 TO 5
   CALL walk(L)
   CALL imadoko(x(n),y(n))
   LET n=n+1
   CALL right(144)
   CALL walk(L)
   CALL imadoko(x(n),y(n))
   LET n=n+1
   CALL left(72)
NEXT i
SET AREA COLOR "red"
MAT PLOT AREA : x,y
END
```

2.3 番外編: 素朴に頂点座標を読み取って

百円ショップでコンパスと分度器を買って、ネットで方眼紙を入手して、実際に星を描いて、座標を読み取って…

それに近いことをやった諸君の先輩がいましたが、DRAW GRID を使ってグリッドを描いておくと、方眼紙なしでも、一応画面上で割と簡単に作業出来るようですね。

図 6: タートルグラフィックスで星を描く

naivestar.BAS

```
REM naivestar.BAS --- 素朴に座標を与えて星を描く
DIM x(10),y(10)
LET x(1)=4.8
LET y(1)=1.5
LET x(2)=1.1
LET y(2)=1.55
LET x(3)=0
LET y(3)=5
LET x(4)=-1.1
LET y(4)=1.55
LET x(5)=-4.75
LET y(5)=1.5
LET x(6)=-1.8
LET y(6)=-0.6
LET x(7)=-2.9
LET y(7)=-4
LET x(8)=0
LET y(8)=-1.9
LET x(9)=2.9
LET y(9)=-4
LET x(10)=1.8
LET y(10)=-0.6
SET WINDOW -5,5,-5,5
SET AREA COLOR "red"
MAT PLOT AREA : x,y
END
```

目標を緩くしておくと、色々なやり方が出て来て面白い。

昔、プログラミングのテストで、与えられた数を 10 倍にするという問題があって (掛け算命令のない CPU に対する機械語プログラム)、普通は桁ずらしとか使うものだけど (2 進法で 1 桁ずらすと 2 倍、3 桁ずらすと 8 倍、足すと 10 倍とか)、足し算を 10 回繰り返すというプログラムを書いた人がいて、「斬新だ」と評価されたという話 (笑い話？) があります。