情報処理 2 第 5 回

十進BASIC (2) 繰り返しのある計算

桂田 祐史 (かつらだ まさし)

2013 年 5 月 15 日

この授業用の WWW ページは http://www.math.meiji.ac.jp/~mk/syori2-2013/

今日は、コンピューターによる計算で最も大事な要素である (と私が信じている) 「繰り返しのある計算」のプログラミングを取り上げます。同様のことを C 言語のプログラムでやったはずですが、気持ちを新たに「これくらいのプログラムはいつでも自力で書けるようになってやる」と考えてチャレンジして下さい。

レポート課題 4、kadai4.BAS または kadai4.pdf を送るように指示していますが、そうでないもの (first.BAS とか) を送った人が結構います。指示は守って下さい。

1 FOR 〜 NEXT による繰り返し

コンピューターが最もその能力を発揮するのは、**繰り返し計算**をしているときではないでしょうか。人間が書く短いプログラムで大規模な計算 (データ処理を含む) をするのは、プログラムのどこかで繰り返し処理があるからです。

ここでは、数学の問題を繰り返し処理を用いて解くプログラムを考えます。すべてとはいいませんが、かなりの部分は**漸化式**による計算 (をプログラムにうつしたもの) になります。

コンピューターに繰り返し処理をさせるには、“**再帰的手続き**” などもありますが、ここではポピュラーな繰り返し構文 “FOR 〜NEXT” を使ってみます[1]。

1.1 FOR NEXT 構文

十進 BASIC には繰り返しを記述するための命令が色々ありますが、多分 FOR NEXT 構文が一番よく使われるでしょう。

[1] 十進 BASIC には、他に DO ... LOOP, DO WHILE ... LOOP, DO UNTIL ... LOOP, DO ... LOOP WHILE, DO ... LOOP UNTIL などの繰り返し構文があります。

nenbutsu.bas

```
REM nenbutsu.BAS
REM 単純な繰り返し
FOR n=1 to 100
  print "南無阿弥陀仏"
NEXT n
END
```

形式上は C 言語の for 文に似ています (実は C 言語の for の方が高機能で、全然違う、とも言えますが)。

やってみよう FOR, PRINT, END 等の命令の説明をオンライン・ヘルプで読んでみましょう。(最初、REM を調べようと書いておいたのですが、オンライン・ヘルプには載っていませんね。これは注釈行 (remark) といって、実行には関係無い、メモ書きをするための行を作るためのものです。C99 言語の // に近いです。)

1.2 簡単な漸化式

例題 1. 定数 r が与えられたとき、漸化式

$$a_1 = 1, \quad a_n = ra_{n-1} \quad (n \geq 2)$$

で定義される数列 $\{a_n\}_{n\in\mathbf{N}}$ の最初の 100 項を求めよ。— 要するに等比数列 $a_n = r^{n-1}$ ですが、この仕事を漸化式の通りに遂行するには、例えば次のようなプログラムを書けば OK です (上の漸化式とプログラムを見比べて下さい)。

配列を使うバージョン touhi1.bas

```
REM touhi1.BAS
REM 等比数列（配列を使うバージョン）
DIM A(100)
INPUT PROMPT "r=": r
A(1)=1
print 1;A(1)
FOR n=2 to 100
  A(n)=r*A(n-1)
  print n;A(n)
NEXT n
END
```

(細かい注: 十進 BASIC の配列は、普通添字が 1 から始まります。つまり DIM A(100) とすると、A(1), A(2), A(3), ..., A(100) が使えるようになります。OPTION BASE 0 とすると、添字は 0 から始まります。DIM A(2 TO 8) のように、添字の下限と上限を指定することも出来ます。)

やってみよう DIM, INPUT PROMPT 等の命令の説明をオンライン・ヘルプで読んでみましょう。

等比数列はものすごい勢いで大きくなったり小さくなったりするので、演算精度を上げておくべきかもしれません。これは (1000 桁演算モード) ボタンを押しても実現できますが、プログラムの先頭部分に例えば

1000 桁モードで計算する (ボタンを押さなくても) ことを宣言

```
OPTION ARITHMETIC DECIMAL_HIGH
```

と書いてもよいでしょう。

ちょっと質問 “arithmetic” って何でしょうね？[2]単語の意味を確認する習慣を身につけることをお勧めします。“decimal” は分かりますか？

実は一度 `A(n-1)` として使われた後はもう使われなくなるので、次のようなプログラムで済ませることが出来ます。

配列を使わないバージョン touhi2.bas

```
REM touhi2.BAS
REM 等比数列 (配列を使わないバージョン)
INPUT PROMPT "r=": r
A=1
print 1;A
FOR n=2 to 100
  A=r*A
  print n;A
NEXT n
END
```

配列変数の代りに、普通の変数 1 個 `A` だけで済んでいる理由を理解しましょう。代入文 `A=r*A` は、数学に現れる等式とは違って、一般には

変数名 = 式

と表される (代入文の) 文法に従ったもので、(1) 最初に `=` の右にある式を評価し、(2) その値を `=` の左にある変数に記憶します。例えば `A=A+1` は `A` の値を 1 増やすコードです。

1.3 レポート課題 5A (必修)

フィボナッチ数列は、漸化式

$$a_1 = 1, \quad a_2 = 1, \quad a_n = a_{n-1} + a_{n-2} \quad (n \geq 3)$$

で定義されるが、この最初の 100 項までを計算するプログラムを作成して、実行し、第 100 項 a_{100} が何か答えよ。実行結果 (a_1 から a_{100} の値が表示されている) `kadai5a.TXT` を付録として文書に取り込むこと。

これは出席がわりで、出来る限り今日中に済ませること。

[2]arithmetic (1) 算数、算術 (2) 算数の能力; 計算; decimal arithmetic 十進算, mental arithmetic 暗算

プログラムの作り方自体は、前項の touhi1.BAS, touhi2.BAS が参考になります。

もしできれば配列を使わないプログラム kadai5a2.BAS も作成してみると良いでしょう (ちゃんと動くかどうか実行結果を比べること)。

例えばこんなふうにして下さい

```
\documentclass[12pt]{jarticle}
\usepackage[a4paper]{geometry}% 好みの問題
\usepackage{amsmath,amssymb}% 今回は \Bbb で amssymb を使っている
\usepackage{moreverb}% 今回これが必要

\newcommand{\N}{{\Bbb N}}
\begin{document}
\title{情報処理2 課題5A レポート}
\author{2年16組99番 数学 学}
\date{2013年5月15日}
\maketitle

\section{課題5A}

漸化式
\[
  a_1=1,\quad a_2=1,\quad a_{n}=a_{n-1}+a_{n-2}\quad\text{($n\ge 3$)}
\]
で定義されるフィボナッチ数列 $\{a_n\}_{n\in\N}$ の、
最初の 100 項を十進BASICで計算し、第100項目を答える。

\section{プログラム}

\listinginput{1}{kadai5a.BAS}% これで kadai5a.BAS を取り込みます。

実行した結果から、第100項目は
\[
  a_{100}=
\]

\appendix% ここから付録（節の名前が A,B,.. になる）

\section{プログラムの実行結果}

kadai5a.BAS の実行結果は次のようになる。

\verbatimtabinput{kadai5a.TXT}% これで kadai5a.TXT を取り込みます。

\end{document}
```

こんな感じ[3] になります。

注意　プログラムによっては、実行結果が膨大になり、それをレポートにそのまま含めるのは不適切です (特に紙で提出させていた頃は資源の無駄遣いも発生)。ここでは結果が 100 行程度なので、そのまま送ってしまっても構いません。

(念のため) コマンドプロンプトでこんなふうに処理

```
Z:¥.windows2000¥syori2>platex kadai5a.tex
Z:¥.windows2000¥syori2>dviout kadai5a.dvi
Z:¥.windows2000¥syori2>dvipdfmx kadai5a.dvi
```

こうして出来た kadai5a.pdf を Oh-o! Meiji のレポートシステムで提出。

1.4　和 $\sum$ の計算

一見漸化式と関係なさそうでも、漸化式を用いて計算すると便利というものが結構あります。

例えば数列 $\{a_j\}$ の第 1 項から第 n 項までの和 $s = \sum_{j=1}^{n} a_j$ は、部分和

$$s_j := \sum_{k=1}^{j} a_k$$

を導入すると、数列 $\{s_j\}$ の第 n 項である、すなわち $s = s_n$ ですが、$\{s_j\}$ は

$$s_1 = a_1, \quad s_j = s_{j-1} + a_j \quad (j \geq 2)$$

あるいは

$$s_0 = 0, \quad s_j = s_{j-1} + a_j \quad (j \geq 1)$$

という漸化式で定義することができます。例えば

$\sum_{j=1}^{n} a_j$ の計算

```
s=0
for j=1 to n
  s=s+(a_j を計算する式)
next j
print s
```

のようなコード[4]で s_n が計算できます。

[3] http://www.math.meiji.ac.jp/~mk/syori2-2013/kadai5a.pdf

[4] コンピューター・プログラム (特にその断片) のことをコードと言うことがあります。

suaresum.BAS

```
REM squaresum.BAS
REM 自然数の平方の和
INPUT PROMPT "N を入力してください:": N
S=0
FOR J=1 to N
  S=S+J^2
NEXT J
PRINT S
REM 知っている公式で値を計算して確認
PRINT N*(N+1)*(2*N+1)/6
END
```

1.5 レポート課題 5B

自然数 n が与えられたときに

$$s_n := \sum_{j=1}^{n} \frac{1}{j}, \quad t_n := \sum_{j=1}^{n} \frac{1}{j^2}, \quad r_n := \sum_{j=1}^{n} \frac{1}{j^2+j}$$

を計算するプログラムを書き、$n = 1, 10, 100, 1000, \ldots$ のとき[5]、値がどうなるか調べて (記録を取ること — 紙に書いても良いですが、COPY & PASTE してコンピューター上に残すのが間違いが起きづらいでしょう)、$\{s_n\}, \{t_n\}, \{r_n\}$ が収束するか、発散するか、予想して下さい。

なお、$n \to \infty$ のときの極限については、実は (数値計算しなくても) ある程度分かるはずです (収束・発散の区別については基礎数学 4 までの知識で判断出来るはず)。なるべく、そのことを踏まえて結果を説明して下さい (確かに収束しそうだとか、発散しそうだとか、もし極限値が分かっているならば第 n 項と極限値との差を計算してみる等)。

もっとも、この講義は、解析学ではないので、極限が分からなければ、分からないでも構いません。必要なのは、計算結果をきちんと示すことと、その結果から数列が収束するのか、発散するのか、収束する場合の極限値の見通しをつけることです。

レポートは TEX を使って `kadai5b.pdf` という名前の PDF ファイルを作成し、Oh-o! Meiji のレポート提出システムで送って下さい。締め切りは、6 月 4 日 (火曜) 18:00 とします。

TEX 文書は以下のような感じになるでしょう

[5] n が大きくなると計算に時間がかかります。概算でよければ精度は無暗に高くしないことを勧めます。十進 BASIC の場合は 1000 桁演算モードでない、普通の演算モードで計算すると速いでしょう。一方、有理数演算モードにしても面白いかも。どれくらいスピードに差が出るでしょうか? 色々試してみてください。

kadai5b.tex

```
\documentclass[12pt]{jarticle}
\usepackage[a4paper]{geometry}% 好みの問題
\usepackage{amsmath,amssymb}% 今回は不要かも
\usepackage{moreverb}% 今回これが必要
\begin{document}
\title{情報処理２ 課題5Bレポート}
\author{2年16組99番 数学 学}
\date{2013年5月15日}
\maketitle

\section{プログラム}

次のプログラムで $n=1,10,100,1000,10000$ のときの $s_n$, $t_n$, $r_n$ の値が
計算できます。

\listinginput{1}{kadai5b.BAS}% これで kadai5b.BAS を取り込みます。
                             % 複数あれば、同じように取り込めば良いでしょう。
\section{プログラムの実行結果}

kadai5b.BAS の実行結果は次のようになる。

\verbatimtabinput{kadai5b.TXT}% kadai5b.TXT を取り込みます。
% 必要ならば複数の実行結果を取り込めば良い。
% コピー＆ペーストで結果をまとめても構いません。
% その場合は verbatim 環境を使うのが簡単でしょうか。

\section{結果の分析}
(以下略)

\end{document}
```

工夫のヒント: 工夫すると、1つのプログラムで複数の n の値に対する s_n, t_n, r_n の値を一気に計算することができます。もし無理なくできるならば、そういうプログラムを作ってみて下さい (加点要素になります)。

2 研究課題1 — アルキメデスの円周率計算

これはレポートを提出するかどうか任意 (余裕がある人向けの「挑戦課題」)。締め切りはこの講義の最終回まで。提出方法は、syori2 @ math.meiji.ac.jp (@は ASCII の@) に電子メールを送ること。

半径 1/2 の円の円周 (の長さ) は、円周率 π に等しい ($r = 1/2$ なので $2\pi r = 2\pi \cdot 1/2 = \pi$)。有名なシラクサのアルキメデス (BC287?〜BC212) は、内接正 n 角形の周の長さ p_n と、外接正 n 角形の周の長さ q_n を考え、

$$p_n < \pi < q_n, \quad \lim_{n\to\infty} p_n = \lim_{n\to\infty} q_n = \pi$$

をもとにして、実際に $n = 6, 12, 24, 48, 96$ に対して p_n, q_n を計算して (正六角形から始めて、辺の数を倍にしていった)、π の評価を得た (「円の測定」[6] という短い論文の中で、$3+\frac{10}{71} < \pi < 3+\frac{1}{7}$ を証明している)。アルキメデスは $\{p_n\}, \{q_n\}$ について成り立つ漸化式 (?)

$$q_{2n} = \frac{2p_nq_n}{p_n + q_n}, \quad p_{2n} = \sqrt{p_nq_{2n}}$$

を用いて計算したという[7]。$n = 3 \cdot 2^k$ の場合、すなわち内接・外接正 $3 \cdot 2^k$ 角形の周長をそれぞれ P_k, Q_k とおく:

$$P_k := p_{3\cdot 2^k}, \quad Q_k := q_{3\cdot 2^k}.$$

このとき $\{P_k\}, \{Q_k\}$ について成り立つ漸化式を導き、

$$P_1 = p_6 = 3, \quad Q_1 = q_6 = 2\sqrt{3}$$

と合わせて用いて $\{P_k\}, \{Q_k\}$ を計算するプログラムを作成し、以下の (1), (2) に答えよ。

(1) 正 96 角形 ($k = 5$ の場合) を利用すると、π のどのような評価が得られるか。

(2) ネットで検索すると、内接正多角形の周長で円周率の近似値を求めた歴史上の話が色々見つかる (円周率マニアは多く、歴史の蘊蓄(うんちく)を語ってくれます)。それらの話をいくつか選んで確認せよ (正 n 角形の周の長さを用いて小数点以下○○桁の値を求めたとある場合、本当にそれができるか — ときどきウソが書いてある)。

(3) P_kQ_k の 1 : 2 または 2 : 1 の比の内分点は、π のより高精度な近似値となる。そのことを確めよ。(どうしてそうなるか説明できたら素晴らしい。)

以下はヒント。

- もし高精度計算が必要な場合は、`OPTION ARITHMETIC DECIMAL_HIGH` と宣言して、1000 桁演算を行うとよい。
- 十進 BASIC には、円周率の近似値を持つ `PI` という定数が事前に定義されているので、チェックに利用できる。

[6] `http://en.wikipedia.org/wiki/Measurement_of_a_Circle`

[7] 現代の数学を使えば、$p_n = n\sin(\pi/n)$, $q_n = n\tan(\pi/n)$ と表される。これは sin, tan の計算には直接は役立たないが、漸化式が成立することの確認には役立つかもしれない。

3　研究課題 2 — Briggs の方法による対数計算

これもレポートを提出するかどうか任意 (余裕がある人向けの「挑戦課題」)。締め切りはこの講義の最終回まで。提出方法は、syori2 @ math.meiji.ac.jp (@は ASCII の@) に電子メールを送ること。正数 $a(\neq 1)$, b に対して、対数 $\log_a b$ の近似値を以下の手順で計算することができる。

$\log_a b$ の近似値の求め方 (by Briggs)

数列 $\{a_n\}$, $\{b_n\}$ を

$$a_0 := a, \quad b_0 := b,$$
$$a_{n+1} := \sqrt{a_n}, \quad b_{n+1} := \sqrt{b_n} \quad (n = 0, 1, 2, \dots)$$

で定め、十分大きな n に対して

$$\frac{b_n - 1}{a_n - 1}$$

を $\log_a b$ の近似値に採用する。

Henry Briggs (1561–1630) は、歴史上初めての常用対数表を、この方法を用いて作成した (彼は $n = 54$ を採用したという。ハイラー＆ヴァンナー, 解析教程 上, シュプリンガーフェアラーク東京 (1997) に証明がある。対数表は、常用対数でなければ John Napier 1550–1617) が作成したもの (1614) が最初である。)。

- 十進BASICで10進1000桁で計算するには、プログラムの先頭付近で `OPTION ARITHMETIC DECIMAL_HIGH`
- 十進 BASIC では、$\sqrt{a}$ は `SQR(A)` で計算できる。

以下の (1), (2) に答えよ。

(1) Briggs の方法で $\log_{10} 2$ を求め、

$$\log_{10} 2 = 0.30102\,99956\,63981\,19521\,37388\,94724\,49302\,67681\dots$$

と比較せよ (精度はどの程度か)。

(2) なぜこの方法で log が計算できるか説明せよ (Briggs は微積分のない時代の人だが、微積分を使って説明しても構わない— 高校数学レベルなのでおじけずに挑戦して下さい)。

微積分の教科書の古典である高木貞治『解析概論』岩波書店に、対数を計算する話題 (Briggs よりはずっと近代的) が載っている。それを試してみるのも面白いかも知れない。

4　研究課題 3 — $\tan z$ の Taylor 展開

これもレポートを提出するかどうか任意 (余裕がある人向けの「挑戦課題」)。締め切りはこの講義の最終回まで。提出方法は、syori2 @ math.meiji.ac.jp (@は ASCII の@) に電子メールを送ること。

微積分の教科書を見ると、多くの初等関数 (exp, cos, sin, $(1+x)^\alpha$, $\log(1+x)$ など) の Taylor 展開が載っていますが、tan の Taylor 展開が載っている本はあまりありません。実は tan の Taylor 展開の一般項の係数は、大学 2 年生が知っているものを使って表すことは出来ません (Bernoulli (ベルヌーイ) 数というものを使うと表示できます)。しかし、Taylor 展開の最初の数項を求めるだけならば、大学 1,2 年生レベルの数学で十分です (素朴で良ければ、$f(x) = \tan x$ を微分して $f^{(n)}(0)$ を求めていけば OK — でもこれを十進 BASIC で実行するのは至難の技)。ここでは『関数論 1』で学ぶ事項を利用して計算する方法を考えてみます。

$\cos z$, $\sin z$ は $\mathbf{C}$ 全体で正則で、$\cos z \neq 0$ ($|z| < \pi/2$) であるから、

$$F(z) := \tan z = \frac{\sin z}{\cos z}$$

は $|z| < \dfrac{\pi}{2}$ で正則であり、

$$F(z) = \sum_{n=0}^{\infty} c_n z^n \quad (|z| < \frac{\pi}{2})$$

と Taylor 展開が出来るはずである。これを求める (n が小さい方から c_n の値をいくつか求める) ことを考えよう。

補題 4.1 (幕級数の割り算) 一般に、0 の近傍における収束冪級数

$$f(z) = \sum_{n=0}^{\infty} a_n z^n, \quad g(z) = \sum_{n=0}^{\infty} b_n z^n, \quad F(z) = \sum_{n=0}^{\infty} c_n z^n$$

に対して、$F(z) = \dfrac{g(z)}{f(z)}$ という関係があるとき (暗に $f(0) \neq 0$ を仮定している)、

(1)
$$c_0 = \frac{b_0}{a_0}, \quad c_n = \frac{b_n - \displaystyle\sum_{k=1}^{n} a_k c_{n-k}}{a_0} \quad (n \in \mathbf{N})$$

が成り立つ。

証明 $g(z) = f(z)F(z)$ であるから、絶対収束級数について成り立つ公式

$$(A_0 + A_1 + A_2 + \cdots)(B_0 + B_1 + B_2 + \cdots) = A_0B_0 + (A_0B_1 + A_1B_0) + (A_0B_2 + A_1B_1 + A_2B_0) + \cdots$$

すなわち

$$\left(\sum_{n=0}^{\infty} A_n\right)\left(\sum_{n=0}^{\infty} B_n\right) = \sum_{n=0}^{\infty}\left(\sum_{k=0}^{n} A_k B_{n-k}\right)$$

から、

$$f(z)F(z) = \left(\sum_{n=0}^{\infty} a_n z^n\right)\left(\sum_{n=0}^{\infty} c_n z^n\right) = \sum_{n=0}^{\infty}\left(\sum_{k=0}^{n} a_k c_{n-k}\right) z^n.$$

これが $g(z)=\sum_{n=0}^{\infty} b_n z^n$ と等しいので、係数を比較して、

$$\begin{aligned} b_0 &= a_0c_0, \\ b_1 &= a_0c_1 + a_1c_0, \\ b_2 &= a_0c_2 + a_1c_1 + a_2c_0, \\ &\vdots \qquad\qquad \vdots \\ b_n &= a_0c_n + \sum_{k=1}^{n} a_k c_{n-k} \quad (n \in \mathbf{N}) \end{aligned}$$

が得られるから、上から

$$c_0 = \frac{b_0}{a_0}, \quad c_1 = \frac{b_1 - a_1c_0}{a_0}, \quad c_2 = \frac{b_2 - a_1c_1 - a_2c_0}{a_0}, \quad \cdots, \quad c_n = \frac{b_n - \sum_{k=1}^{n} a_k c_{n-k}}{a_0} \quad (n \in \mathbf{N}).\blacksquare$$

次の問に答えよ。
上の公式 (1) と、cos, sin の 0 のまわりの Taylor 展開

$$\cos z = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} z^{2n}, \quad \sin z = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} z^{2n+1} \quad (z \in \mathbf{C})$$

を用いて、$\tan z$ の 0 のまわりの Taylor 展開を 10 次の項まで求めよ。

- 十進 BASIC で有理数計算をするには、プログラムの先頭付近で `OPTION ARITHMETIC RATIONAL` と宣言する。
- cos, sin の Taylor 展開の係数は漸化式で計算するのが簡単だが (これについては次回以降詳しく解説する予定)、冪乗演算子 `^` や、階乗を計算する関数 `FACT()` を用いても良い。
- 検算用に
$$\tan z = z + \frac{1}{3}z^3 + \frac{2}{15}z^5 + \frac{17}{315}z^7 + \cdots$$
を掲げておく。
- $\tan z$ は奇関数なので、偶数次の項は現れない。このことを用いると多少効率が上がるが、それはしてもしなくても良い。