## 7.3 自然対数の底

ああ、長い間地面を這い回って泥にまみれて疲れただろう。TEX にもずいぶん苦労をかけた。でも、最後にもうひと働きしてもらおう。途中、$\left(1+\dfrac{1}{n}\right)^n$ の計算を少しだけやったのを覚えているだろうか。この値は一定の値に収束する。配列表示も手にしたことなので、ここで宿題になっていた値を計算させてみよう。ただし、別の式を用いるけど。

まず、$\left(1+\dfrac{1}{n}\right)^n$ は一定の値 $2.71828\cdots$ に収束することが知られていて、それは**自然対数の底**と呼ばれる定数である。そして円周率が $\pi$ で表されるように、自然対数の底も $e$ という姿をとる。もう少し数学っぽい書き方をすると

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e \quad (= 2.71828\cdots)$$

である。これとは別の、$e$ を求める計算式もある。こちらも数学っぽい書き方で

$$\lim_{n\to\infty}\left\{\frac{1}{0!}+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\cdots+\frac{1}{n!}\right\} = e$$

である。どちらも極限値としては同じ値 $e$ へ収束するが

$$\left(1+\frac{1}{n}\right)^n \neq \frac{1}{0!}+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\cdots+\frac{1}{n!}$$

である。有限の項で打ち切っても等しい値にならないことは、たとえば $n=2$ の場合の

$$\left(1+\frac{1}{2}\right)^2 = 2.25、\qquad \frac{1}{0!}+\frac{1}{1!}+\frac{1}{2!} = 2.5$$

からも分かる。あくまでも同じ値に収束する、異なる式ということである。

式だけを見れば左辺の方が簡単そうだが、実際にマクロを書くとなると右辺の方が易しい。そこで、右辺を計算式に用いて収束の様子を見ることにしよう。

```
\newcount\dwn \newcount\zero
\newcommand\arraydivide{\ARY=10
  \loop \ifnum\ARY>0
  \r=\ifcase\ARY%case0 is null
    \or\i  \or\ii  \or\iii  \or\iv \or\v
    \or\vi \or\vii \or\viii \or\ix \or\x\fi
  \dwn=\r \divide\r\n
  \ifcase\ARY%case0 is null
    \or\i  \or\ii  \or\iii  \or\iv \or\v
    \or\vi \or\vii \or\viii \or\ix \or\x\fi=\r
  \multiply\r\n \advance\dwn-\r \multiply\dwn10000
  \advance\ifcase\ARY%case0 is null
    \or\zero \or\i  \or\ii  \or\iii  \or\iv
    \or\v    \or\vi \or\vii \or\viii \or\ix\fi\dwn
  \advance\ARY-1 \repeat
}
```

そのためには、配列で計算できる割り算のルーティンが必要になるので用意した。この除算処理の仕方は、配列による加算マクロ\arrayadvance と似通っている。ただし除算では、繰り上がり処理をするところで繰り下がり処理をする必要がある。そのための変数\dwn と配列\zero を新たに追加したが、他の配列変数はまったく同じものを使っている。

では、マクロ\arraydivide を利用して計算してみよう。

```
\makeatletter
\newcommand\evalue[1]{%
  \newcount\n \newcount\r
  \TOP=1 \x=10000 \n=1
  \@whilenum\n<#1 \do{%
    \arraydivide \arrayadvance
    \advance\n1
  [\number\n] \arraydisplay\par}
}\makeatother
```

これで\evalue{36}を処理すると

[1] 2.0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

[2] 2.5000 0000 0000 0000 0000 0000 0000 0000 0000 0000

[3] 2.6666 6666 6666 6666 6666 6666 6666 6666 6666 6666

[4] 2.7083 3333 3333 3333 3333 3333 3333 3333 3333 3332

[5] 2.7166 6666 6666 6666 6666 6666 6666 6666 6666 6665

[6] 2.7180 5555 5555 5555 5555 5555 5555 5555 5555 5553

[7] 2.7182 5396 8253 9682 5396 8253 9682 5396 8253 9679

[8] 2.7182 7876 9841 2698 4126 9841 2698 4126 9841 2694

[9] 2.7182 8152 5573 1922 3985 8906 5255 7319 2239 8584

[10] 2.7182 8180 1146 3844 7971 7813 0511 4638 4479 7173

[11] 2.7182 8182 6198 4928 6515 9531 8261 9849 2865 1590

[12] 2.7182 8182 8286 1685 6394 6341 7241 1950 1897 2791

[13] 2.7182 8182 8446 7590 0231 4557 8701 1342 5668 9806

[14] 2.7182 8182 8458 2297 4791 2287 5948 2727 7366 9592

[15] 2.7182 8182 8458 9944 6428 5469 5764 7486 7480 1577

[16] 2.7182 8182 8459 0422 5905 8793 4503 2784 1862 2326

[17] 2.7182 8182 8459 0450 7051 6047 7958 4860 5061 1781

[18] 2.7182 8182 8459 0452 2670 8117 4817 1086 9683 3417

[19] 2.7182 8182 8459 0452 3492 8752 7283 3519 9400 2976

[20] 2.7182 8182 8459 0452 3533 9784 4906 6641 5886 1453

[21] 2.7182 8182 8459 0452 3535 9357 4317 2980 7147 3761

[22] 2.7182 8182 8459 0452 3536 0247 1108 6905 2204 7047

[23] 2.7182 8182 8459 0452 3536 0285 7925 7075 8511 5450

[24] 2.7182 8182 8459 0452 3536 0287 4043 0832 9607 6633

[25] 2.7182 8182 8459 0452 3536 0287 4687 7783 2451 5080

[26] 2.7182 8182 8459 0452 3536 0287 4712 5742 8714 7327

[27] 2.7182 8182 8459 0452 3536 0287 4713 4926 5613 3706

[28] 2.7182 8182 8459 0452 3536 0287 4713 5254 5502 6076

[29] 2.7182 8182 8459 0452 3536 0287 4713 5265 8602 2364

[30] 2.7182 8182 8459 0452 3536 0287 4713 5266 2372 2240

[31] 2.7182 8182 8459 0452 3536 0287 4713 5266 2493 8365

[32] 2.7182 8182 8459 0452 3536 0287 4713 5266 2497 6368

[33] 2.7182 8182 8459 0452 3536 0287 4713 5266 2497 7519

[34] 2.7182 8182 8459 0452 3536 0287 4713 5266 2497 7552

[35] 2.7182 8182 8459 0452 3536 0287 4713 5266 2497 7552

が出力される。マクロ`\evalue`は引数に $n$ を与えると、$\dfrac{1}{0!}$ 乗から $\dfrac{1}{(n-1)!}$ 乗までの総和を出力するようにした。この場合は $n=34$、35 で同じ値が出力されるので、ここが着地点であることが分かる。

マクロ`\evalue`の仕組みはこうだ。最初の割り算の結果を配列`\x`〜`\i`に入れたまま割り算を進めると、`\x`〜`\i`のつながりは、順に $\dfrac{1}{1}$、$\dfrac{1}{1\cdot 2}$、$\dfrac{1}{1\cdot 2\cdot 3}$、$\ldots$ の 40 桁の値を表すものになる。そのまま`\X`〜`I`へ加算すれば、自然と $\dfrac{1}{1}+\dfrac{1}{2!}+\dfrac{1}{3!}+\cdots$ が求められる。先頭の $\dfrac{1}{0!}$ は 1 であるから、あらかじめ最上位の桁に格納しておいた。そして $\dfrac{1}{35!}$ 以降を計算させる意味はないから、`\evalue{36}`を実行して $\dfrac{1}{35!}$ まで加算したのである。理由は、$\dfrac{1}{35!}$ が小数点以下 40 桁まで 0 が続く数だからである。

なぜそのことが分かるのだろう。それは、あらかじめ $n!$ が 40 桁を超える場合を計算したからだ。計算の仕方はこうである。$n!>10^{40}$ となる $n$ が分かればよいので、不等式を解くために両辺とも 10 の対数をとって

$$\log_{10} n! > \log_{10} 10^{40}$$

を考える。$n! = n(n-1)(n-2)\cdots 1$ と対数の性質 $\log_{10} MN = \log_{10} M + \log_{10} N$ から、不等式は

$$\log_{10} n + \log_{10}(n-1) + \log_{10}(n-2) + \cdots + \log_{10} 1 > 40$$

となる。さすがにこれを簡単に解くわけにはいかないが、対数表と電卓が用意できれば大して時間をかけずに計算できる。でも、表計算ソフトで計算する方が楽だろう。実際にこれを満たす $n$ を調べると、$n = 35$ で和が 40.01423265 になることが分かる。したがって、40 桁の精度の計算では $\dfrac{1}{35!}$ は 0 と同等なのである。

ということは、本来足されるべき $\dfrac{1}{35!}$ 以降の値が反映されないので、いま出力された $e$ の近似値は精確な値より若干小さいはずである。実際、最後の 2 桁分—つまり 39, 40 桁—は正しくない。

さあ、TEX でこれだけの計算ができるなら当然 $\pi$ の近似値も計算できるね。$\pi$ の計算には、さらに配列で引き算を行うルーティンが必要だ。この場合は負の数を足すわけにはいかないから、配列による加算ルーティンは使えない。よって新たに配列による減算ルーティンを書く必要がある。

でも、蝶道を這い回るのはそろそろ終りにしよう。$\pi$ の近似値を計算しようとすれば、マクロ `\evalue` より複雑で長いコードを書くことになるけれど、決してできないことではない。この先の道は君たちに開拓してもらおう。