3. . . の蝶道

3.1　円周率

円周率は不思議なもので、古くから多くの人々を魅了してきた。いちばん簡単に円周率を表す数は 3 である。また、もっとも効率のよい近似値なら 3.14 というところだろうか。円周率は、小数点以下に不規則な数が並ぶので、昔から有効桁数を高める競争が行われてきた。コンピュータが発達した現代でも、それは続けられている。

円周率を求める古典的な方法は、アルキメデス[1]が考案した。円に内接する正 n 角形の周長と円に外接する正 n 角形の周長を計算し、それらにはさまれた値を円周率とすることである。はさまれた値といっても、ある範囲にはさまれてるわけだから、値が確定しない。したがって、内接 n 角形の周長と外接 n 角形の周長で、一致している部分までが円周率の正しい値を示していることになる。

これらの計算をするには、三角比に関する知識があるとよいのだが、ここではアルキメデスの方法で円周率の近似をすることが目的ではない。期待した諸君には申し訳ないが省略させてもらおう。

円周率はギリシア文字 π で代用される。円周率が通常の分数で表せないので当然の処置だろう。数値計算をするソフトウェアには大抵、円周率の値が組み込まれていて、いつでも自由に呼び出せるものだが、TEX は数値計算をするためのソフトウェアではないから当然そんな値は保持していない。それなのに、TEX で数値計算をしているってどういうこと？

円周率は通常の分数で表せないけれど

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots$$

といった、**無限級数**で表すことは可能だ。これはグレゴリー[2]もしくはライプニッツ[3]の功績だ。式が $\pi = \cdots$ ではなく $\frac{\pi}{4} = \cdots$ と書いてあるのは、その求め方に由来しているが、詳しいことは別の

[1] シラクサのアルキメデス（287?B.C.–212B.C.）：古代ギリシアの数学・物理学者。
[2] ジェームス・グレゴリー（1638–1675）：イギリスの数学者・発明家。
[3] ゴットフリード・ウィルヘルム・ライプニッツ（1646–1716）：ドイツの哲学・数学者。

書物を参考にしてもらいたい。右辺を無限に計算することができるなら、その結果はぴったり円周率の値となる。しかし、これは計算機向きの式ではない。なぜなら**収束**が非常に遅いからだ。収束が遅いことは、計算機の処理速度がいかに速くても致命的なものである。

収束が遅いことは式を見ているだけでも理解できると思う。たとえば級数を 5 億項先まで加えても、分母は 10 億程度の大きさである。これでは小数点以下 8 桁の精度にしかならない。

しかし悲観ばかりしていても仕方ない。収束が遅いことを承知の上で、この方法を TEX で再現しよう。

```
\makeatletter
\newcommand\qpi[1]{%
  \newcount\n \newdimen\p
  \n=1
  \loop
    {\multiply\n2 \advance\n-1 \dimendiv(1, \n)}%
    \dimensum(\strip@pt\p, \ifodd\n\else-\fi\strip@pt\dimen@)%
    \p=\dimen@
    \advance\n1
  \ifnum\n<#1 \repeat
  \multiply\p4 \strip@pt\p
}\makeatother
```

これで、『円周率の近似値は`\qpi{100}`である。』と書けば『円周率の近似値は 3.15149 である。』が出力される。マクロ名は、π の四分の一（quater pi）を求める計算式にちなんだ。

変数は項数である`\n` と円周率の値を格納する`\p` を用意した。それで引数から得た値まで`\loop` をかけると、引数分の項が加算される。そのことは`\repeat` とその上の行から分かるだろう。しかし、加算する値は`\n` 項そのものを使うのでなく、n 番目の奇数であるから $2n-1$ を分母とする逆数値が必要だ。そこで、`\n` を 2 倍して 1 引いて逆数にしている。すると`\n` の値が変化してしまうため、ここはグルーピングしなくてはならない。

しかしながら、加算は正負の数を交互に行われる。そのようなときは、`\if` 命令で分岐させるのが常道だが、ここは TEX の性質を利用して変わったことをしている。それが`\ifodd\n\else-\fi` である。これがなければ、この行は単に加算ルーティン`\dimensum(\strip@pt\p, \strip@pt\dimen@)` であることが見て取れる。じゃあ`\ifodd\n\else-\fi`って何だ？　われわれは数の正負を何で判断するかというと、それは “–” の有無である。TEX も同じだ。`\dimensum` は 2 番目の引数が`-xxx` のとき引き算をする。引き算にしたいときだけ “-” を付ければよいのである。

では、いつ引き算にすればよいのだろう。それは、いま行っている計算は偶数項が負の項であるから、`\n` が偶数のときに “-” が付けばよいことになる。偶数か奇数かを判断するには、以前登場した`\ifodd` が使える。つまり`\ifodd\n\else-\fi` とは、`\n` が奇数なら何もせず、偶数のときだけ

“–” を返す。すなわち`\n`が偶数のときに`\dimensum(\strip@pt\p, -\strip@pt\dimen@)`が実行されるのだ。それが、`\dimensum`の引数が妙な表記になった理由である。この荒技は TEX ならではのものなのだ。もちろん符号を転換するために -1 を掛けてもよいけれど、それでは簡潔な記述は望めない。

さて、`\p`には $\frac{\pi}{4}$ の値が入る。ここで`\p`に値を確保しておかないと、`\dimen@`は別の値になってしまう。以上の計算で求めたのは $\frac{\pi}{4}$ の値である。それを π の値として出力させるには、`\p`を 4 倍しなければならない。

結果を見て分かる通り、100 項分の計算をした割には、よく知られた値 3.14 とは 0.01 も違う。普通のプログラミング言語でも π の計算は大変なのだから、TEX でこの結果だとしても良しとしなければならないだろう。