6 e の値 1000 桁

マクローリン展開によって e の正確な値を知るメドがたったので、早速 **PowerShell** で計算してみよう。精度は π と同じく 1000 桁である。旅の途中のほんのわずかの寄り道であるが、π のスクリプトを知っている肥えた目には、e のスクリプトはどうってことない。なぜなら、級数は常に足し算をするだけだし、次の項は前の項を n で割るだけでよいからだ。項によって足したり引いたりを判断しなくて済むので、手間を大幅に減らせる。要するに、π のスクリプトのどこを削るかという問題なのである。

[ps script]

```
$ARRAY = 100; $FIGS = 10000000000
$e = @(); 0..($ARRAY+1) | foreach {$e += 0}; $e[0] = 1

function Div($n) {
  foreach ($i in 0..$ARRAY) {
    $u[$i+1] = $u[$i+1] + ($u[$i] % $n) * $FIGS
    $u[$i] = [math]::floor($u[$i] / $n)
  }
}

function eInc {
  for ($i = ($ARRAY+1); $i -gt 0; $i--) {
    $e[$i-1] = $e[$i-1] + [math]::floor(($e[$i] + $u[$i]) / $FIGS)
    $e[$i] = ($e[$i] + $u[$i]) % $FIGS
  }
}

$u = @(); foreach ($i in 0..($ARRAY+1)) {$u += 0}; $u[1] = $FIGS
foreach ($i in 1..451) {
  Div $i; eInc
}

$q = @(); foreach ($i in 1..$ARRAY) { $q += ('{0:d10}' -f [long]$e[$i]) }
[string]$e[0] + "."; "$q"
```

スクリプトを見れば、ほとんどが余分なコードを削っただけと気づくはずだ。わずかの違いといえば、配列 `$e` の用意に際して`$e[0] = 1` としたことぐらいである。e のマクローリン展開は $1 + \frac{1}{1!} + \cdots$ で始まっているが、`function Div($n)` が受け持つことができるのは $\frac{1}{1!}$ からである。そこで、最初の 1 だけはあらかじめ配列に与えてある。

計算回数が 451 回であることに触れておこう。1000 桁の精度で計算する場合、$\frac{1}{n!}$ が $\frac{1}{10^{1000}}$ を下回るとこ

ろまで計算すればよい。つまり $n! > 10^{1000}$ を満たす n を求めればよいことになる。これは、両辺の対数をとって $\log n! > \log 10^{1000}$ とするのがよい。すると、対数の性質 $\log MN = \log M + \log N$ を段階的に用いて

$$\log n(n-1)(n-2)\cdot\cdots\cdot 1 > 1000\log 10$$
$$\log n + \log(n-1) + \log(n-2) + \cdots + \log 1 > 1000\log 10$$

と変形できるからだ。こうなると逆に解きづらい気がするけれど、**PowerShell** の力を借りればわけないことである。次のようなスクリプトで解決する。

[ps script]

```
PS C:\Users\Yours > $n = 1; $s = 0
>> while ($s -lt 1000*[math]::log(10)) {
>> $s += [math]::log($n)
>> $n += 1 }
>>
```

これで $\log n! > \log 10^{1000}$ を満たす n が求められた。`$n` と入力してみよう。

[ps script]

```
PS C:\Users\Yours > $n
451
```

これより、計算回数を 451 回に決めたのである。ただし、最後の配列の数桁は誤差を含むので、確実に 1000 桁の値を望むなら、配列をもうひとつ余分に用意しなくてはならない。単に計算回数を増やしても精度には全く影響がないのだから。

さあ、計算してみよう。もし `evalue.ps1` の名前で保存したなら、プロンプトに対して`./evalue` と打ち込む。**PowerShell** の作法は覚えているね。これで 1000 桁の e を手にすることができる。

π の計算のときもそうだったが、スクリプトは 1000 桁計算限定である。もちろん 2000 桁の e を知りたければ、スクリプトの 1 行めを`$ARRAY = 200` に、下から 6 行めを `foreach ($i in 1..809)` にすればよい。809 の値は、あらかじめ $\log n! > \log 10^{2000}$ を **PowerShell** で計算しておくことになるが。

それにしても、違う桁数の e を調べたくなったら、いちいちファイルを書き換えるのは面倒なものである。できれば、プロンプトに対してスクリプトの実行を指示するとき、配列数や計算回数を与えることができれば便利である。そうすれば、スクリプトはもう書き換える必要がなくなるから。

そう思ったら、1 行めを`$ARRAY = $args[0]` に、下から 6 行めを `foreach ($i in 1..$args[1])` に書き換えた上で、プロンプトに対して`./evalue 200 809` と打ち込めばよい。すると、配列数が 200—桁数なら 2000—で、計算回数を 809 回として e の値を表示してくる。もっとも 809 の値は、あらかじめ求めておかなくてはならないけれど。このことは、**PowerShell** では**引数**を自由にとることができ、順に`$args[0]`、

`$args[1]`、...へ渡される仕組みがあるからである。

スクリプトにちょっと面倒な手を加えてよければ、計算回数は引数に与えることはない。なぜなら、計算回数は配列数から求められるからである。すなわち、計算回数を求めるスクリプトを付け加えて、そこで求めた値を `foreach` 文へ渡せばよいからだ。そうすれば、`./evalue 200` だけで 2000 桁の e が表示できる。

この作業はさほど大変ではないが、**PowerShell** に不慣れな場合は大変な仕事になってしまうはずだ。それでも、自分の腕試しと思って挑戦してほしい。こういう少しずつの積み重ねが、君たちをいっぱしのスクリプト書きに育ててくれるのだから。