

5.2 漸化式

では、フィボナッチ数列に戻ろう。

よく数列の n 番目の項を a_n などと表記することがある。すると次の $n+1$ 番目の項は a_{n+1} 、 $n+2$ 番目の項は a_{n+2} という書き方になるだろう。フィボナッチ数列は、直前の 2 項の和が次の項になっているので、一般に

$$a_{n+2} = a_{n+1} + a_n \quad (5.1)$$

という書き方ができる。特に今は 1, 1 から数列を始めているので、 $a_1 = 1, a_2 = 1$ の条件が (5.1) に付け加わることになる。

これだけの条件と式があれば、この先の項が順次計算できるのだ。(5.1) のような形式で与えられる式を漸化式と呼ぶ。数学の世界に限らず、前後の関係は分かっているのだが、そのものズバリを与える式が不明であることは多い。実はフィボナッチ数列もそんなもののひとつだ。数列の前後の関係は (5.1) により明確に分かっている。では、フィボナッチ数列のズバリ第 n 項を求める式は何だろう。これはちょっと複雑な計算をするので、詳しいことは省くが、フィボナッチ数列の第 n 項 a_n は

$$a_n = \frac{1}{\sqrt{5}} \left\{ \left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right\}$$

で与えられる。

実際に計算すると大変だが $n = 1, 2, 3, \dots$ を代入すれば

$$1, 1, 2, 3, 5, 8, 13, 21, 34, \dots$$

が現れるのだ。式には $\sqrt{5}$ があるのに、フィボナッチ数列には一切登場しない点が興味深い。近似値でよければ、 $\{ \}$ 内の初めの項が $(1.618033\dots)^n$ で、後の項が $(0.618033\dots)^n$ であることを利用しよう。 n が大きいほど $(0.618033\dots)^n$ は 0 とみなせるので、第 a_n 項は近似的に $\frac{1}{\sqrt{5}}(1.618033\dots)^n$ と考えてよい。電卓を使えば、 $n = 9$ のときは $a_n \approx 33.994116$ であることが分かる。確かに第 9 項の 34 に一致している。

VBA には漸化式をうまく扱える仕組みが備わっている。以前、フィボナッチ数列を生成するマクロ [Fibonac2.vba] を覚えているね。そして君たちは [Fibonac2.vba] をもとに、整数 n をセルから読み取って n 番目のフィボナッチ数を表示するマクロを作っただろう。え？ 作ってないだって？ だめだなあ、手抜きをしちゃあ。まあ、いいや。ここでは当時とまったく同じ動作をするものを、漸化式から作ってみよう。次のマクロも、整数 n をセルから読み取って n 番目のフィボナッチ数を表示する。

programming list [NthFib.vba]

```
1: Sub Nthfib()  
2: Dim num As Integer  
3:  
4:     num = Sheet1.Cells(1, 1).Value  
5:     Sheet1.Cells(2, 1) = fib(num)  
6: End Sub  
7:  
8: Function fib(n)  
9:     If n > 2 Then  
10:         fib = fib(n - 1) + fib(n - 2)  
11:     Else  
12:         fib = 1  
13:     End If  
14: End Function
```

これは単に、A1 セルに入力された値に対して、その順番のフィボナッチ数を表示するマクロである。例えば 10 が入力してあれば、A2 セルには “55” が表示される。以前のマクロとは少し違った印象を受けるかもしれない。もし、そんな印象を受けたら、それは関数 `fib(n)` のせいである。

マクロの仕組みを説明しておこう。まず先に、マクロ本体である `Sub Nthfib()` を見ていくことにする。これは実質 2 行のプログラムでしかない。

4: 行目で、何番目のフィボナッチ数を特定するための変数 `num` に、A1 セルの値を読み込んだ。変数名は `n` でもよかったのだが、少々味気ないので意味—つまり *number*—が分かるようにしてみただけである。ちなみに、よく顔を出していた `row, col` は *row, column* の意味で使っていた。

`num` に値が代入されると、5: 行目の `Sheet1.Cells(2, 1) = fib(num)` で、`num` 番目のフィボナッチ数が A2 セルに表示されるのである。それは `fib(num)` の値のはずなのだが、一体どのように計算されるのだろうか。

具体的に `num = 4` が与えられたとして処理を追ってみよう。処理は 8: 行目からの関数 `fib(n)` の受け持ちだ。

8: 行目で 4 の入力を受け取った VBA は、5: 行目の命令により、A2 セルに “fib(4)” を代入しなくなっている。

一方 `fib(4)` は、関数 `fib(n)` により計算がされる。このとき `fib(n)` が受け取る 4 は `n` の値として受け取るわけで、それは条件文 `If` の `n > 2` を満たしているから、素直に 10: 行目の `fib(n - 1) + fib(n - 2)` に代入され、`fib(3) + fib(2)` が返される。あれれ？ 今は `fib(n)` の計算のはずだ。なのに `fib(n)` の計算に `fib(n)` を使うのかい？ 疑問はもっともだが、それでよいのである。もやもやを抱えつつも先へ行こう。

`fib(n)` によって `fib(3) + fib(2)` が返された結果 VBA は、5: 行目の命令により A2 セルに “fib(3)+fib(2)” を代入する必要に迫られる。しかし、`fib(3)` は再び関数 `fib()` の 10: 行目の

処理により $\text{fib}(2) + \text{fib}(1)$ にして返されてしまうのだ。

ところが、 $\text{fib}(2)$ の方は $n > 2$ の条件を満たさないので、こちらは 11: 行目の Else 文による処理で 1 が返される。その結果 VBA は “ $\text{fib}(2)+\text{fib}(1)+1$ ” を A2 セルに代入しようとする。

しかしながら、まだ $\text{fib}(2)$ と $\text{fib}(1)$ は関数呼んで計算を続けるので、さらに $\text{fib}(n)$ により 1 と 1 が帰ってくる。これでようやく VBA は “ $1+1+1$ ” を代入すればよいことが分かり、結局 A2 セルには “3” が代入されるのだ。

一見すると詐欺にあったような印象を受けるだろうか。このような呼び出しは再帰呼び出しという。前後関係がはっきりしていて、一般の式を求める必要がないときなどに有効である。ただし、再帰呼び出しは計算量が爆発的に増加してしまう。例えば $\text{fib}(6)$ の計算でも

$$\begin{aligned}\text{fib}(6) &= \text{fib}(5) + \text{fib}(4) \\ &= \{\text{fib}(4) + \text{fib}(3)\} + \{\text{fib}(3) + \text{fib}(2)\} \\ &= \{(\text{fib}(3) + \text{fib}(2)) + (\text{fib}(2) + \text{fib}(1))\} + \{(\text{fib}(2) + \text{fib}(1)) + 1\} \\ &= \dots\end{aligned}$$

のように、 $\text{fib}(6)$ が $\text{fib}(5)$, $\text{fib}(4)$ を呼び、 $\text{fib}(5)$ と $\text{fib}(4)$ がそれぞれ $\text{fib}(4)$, $\text{fib}(3)$ と $\text{fib}(3)$, $\text{fib}(2)$ を呼び、さらにそれぞれが…。何ということだ。呼び出しが倍々に増えているじゃないか。

あまり大きな数を与えるととんでもないことになるから、とりあえず $\text{fib}(40)$ 程度の計算をさせてみよう。さて、君たちのコンピュータは即座に結果を表示してくれただろうか。おそらく即座に結果を返してくれまい。もし、一瞬のうちに結果が返る高速コンピュータを使っているなら、うらやましい限りだ。理由はこうだ。

$\text{fib}(2)$ や $\text{fib}(1)$ は 2 つの fib 関数と呼ぶわけではないので、単純に $\text{fib}(n)$ の計算が 2^{40} 回行われることはない。実際はもっと少なく、 2^{27} 回ほどだ。一口に 2^{27} 回と言うものの、これでも軽く 1 億回を超えている。1 秒間に 1 億回の呼び出しが可能なコンピュータでも 1 秒はかかるので、“即座” に結果が出るわけではない。われわれが普段使っているコンピュータなら、少なくとも数十秒は要するはずだ。再帰呼び出しをうかつに利用するのは要注意である。