5... の夢見鳥

5.1 黄金比

さあ、これから 5... の夢見鳥路が始まる。と言っても 5 から始まるのではなく、浮遊中に 5 と出会うのが、いままでとは違うところだけど。

まず、フィボナッチ数列

$$1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ \ldots$$

に再び登場してもらおう。

以前見学したように、比の値は $\frac{(\text{後の項})}{(\text{前の項})}$ で求めると $1.618033\ldots$ となり、$\frac{(\text{前の項})}{(\text{後の項})}$ で求めると $0.618033\ldots$ であった。覚えているかな。これははきっちり 1 だけ違っている。さて、この面白い性質を持つ比の値は**黄金比**と呼ばれている。一般には、$1.618033\ldots$ の方を黄金比と呼ぶことが多い。

黄金比とは一体どういう数なのだろうか。計算は簡単にできるので確かめてみよう。

まず、$1.618033\ldots$ に収束する真の値を x としよう。あとから求めた比 $0.618033\ldots$ は、x の逆比 $\frac{1}{x}$ のことである。それが x より 1 小さい値に等しいのだから

$$\frac{1}{x} = x - 1$$

が成り立つ。等式は簡単な 2 次方程式となる。両辺に x を掛けて整理すると $x^2 - x - 1 = 0$ だから、解の公式を使えば一発で解ける。フィボナッチ数の比は正の値なので、$x > 0$ の解を求めると

$$x = \frac{1 + \sqrt{5}}{2}$$

であることが分かる。黄金比は**無理数** $\sqrt{5}$ を含む数なのだ。ここに 5 が登場してきた。解は正の数だけに限っているものの、方程式から一つの解しか得られないということは、この性質を満たす数は黄金比だけということである。

ではシミュレーションしてみよう。ここでは $\frac{1}{x} = x - 1$ を移項して

$$x = \frac{1}{x} + 1 \tag{1}$$

で x を探ることにする。そのほうが x の値を直接計算できるからだ。ところで、探るという言葉が気になるね。なぜそのような言葉遣いをしたかは『

```
\begin{luacode*}
function gratio(x)
    for i = 1, 20 do
        x = 1 / x + 1  -- あとで x = x * x +1 に差し替え
        if (i < 4) or (17 < i) then
           tex.print(x .. '\\' .. '\\')
        elseif i == 4 then
            tex.print('\\qquad' .. '\\vdots' .. '\\' .. '\\')
        end
    end
end
\end{luacode*}
```

』のように定義した関数で『\par\noindent\directlua{gratio(3)}』を処理した様子を見てもらいたい。『

1.3333333333333

1.75

1.5714285714286

⋮

1.6180340143331

1.618033978978

1.6180339924824

』のようになるだろう。

本当はズラーっと出力すればよいのだけれど、改行しなければ版面からはみ出すし、改行すれば紙面の無駄遣いになるし ...。というわけで、先読みをして必要なところだけ改行して出力したのだ。だからスクリプトは、変なところで `if` 文を使って分岐している。我ながらまったく嫌なやつだね。

スクリプトは別に新しことはないけど、`luacode*`環境の `tex.print` 文で TEX の命令を書く場合は、エスケープして `\\vdots` のようにする必要があったんだよね。じゃあ改行を意味する `'\\'` をどうエスケープするかというと、`'\\\'` ではエスケープにならない。これははじめの 2 個の `\\` はエスケープされた `\` と解釈されるものの、次の `\` はエスケープすべき後続の文字がないためエ

ラーとなる。ということで '\\' .. '\\' と書いたのである。

でも、実は改行だけなら \par をエスケープして \\par でよいのだ。でもでも、\\par だとインデントが入るんだよね。ここでは入れたくなかったから \\par を使わなかった。でもでもでも、インデント不要なら \\noindent を入れればよいだろうって？　もう、好きにして。

さて、スクリプトを見て、あれ？コンピュータが 2 次方程式を解くんじゃないんだ、と感じただろう。そう、コンピュータは計算をする機械であって、問題を解く機械ではない。問題を解く手順は人間が与えるのだ。では、この手順は何をしているのだろう。どう見ても $\frac{1}{x} = x - 1$ と同等な—つまり両辺に x を掛けた—方程式

$$1 = x^2 - x$$

を解いているのではないね。でも、解である黄金比が求められている。

それなら $1 = x^2 - x$ を移項した式、$x = x^2 - 1$ を使っても同じことだろう。スクリプトの計算式を `x = x * x - 1` に変えてみる。スクリプトにはあらかじめ予告しておいたよ。Lua では -- 以降、行末まではコメント行となるので、動作に影響しない。さあ、準備は整った。関数を走らせよう。今度は巨大な数が出力されるはずだ。

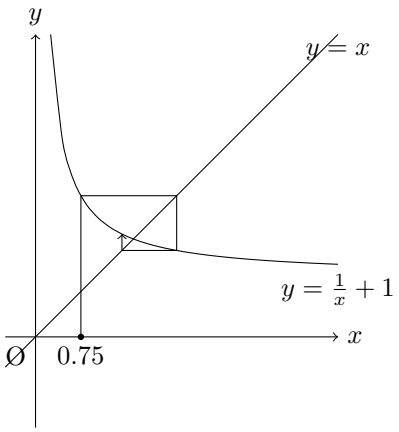
どうして？　同じ関係の方程式を使ったはずなのに。その秘密はグラフを描けば見えてくる。はじめのスクリプトに使ったのは $x = \frac{1}{x} + 1$ で、次のスクリプトに使ったのは $x = x^2 - 1$ だから、それぞれ

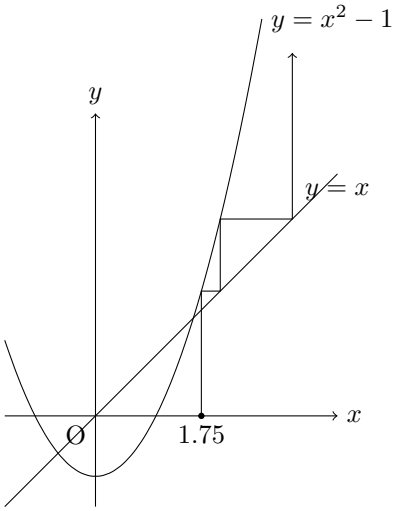
$$y = x \quad と \quad y = \frac{1}{x} + 1$$
$$y = x \quad と \quad y = x^2 - 1$$

のグラフの交点を求めていることになる。ただし、スクリプトは一気に交点を求めているのではない。適当な x から始めて、いわば二つのグラフを渡り歩くような処理がスクリプトのしていることだ。

でも、これ以上詳しいことは言わないでおこう。実際にグラフを描いて、x 軸上の点から始めて、二つのグラフを縦横縦横 ... 交互に進んでみよう。$y = x$ と $y = \frac{1}{x} + 1$ のグラフでは、必ず交点にたどり着くだろう。しかし、$y = x$ と $y = x^2 - 1$ のグラフでは、始めの x 次第で交点にたどり着いたりあさっての方向へ跳んでしまったりするはずだ。こうなったときが、収束しない場合なのである。

その様子を描いておく[1)]。





図には描かなかったが、$y = x$ と $y = x^2 - 1$ のグラフにおいて、出発点をたとえば $(0.75,\ 0)$ から始めると反対側の交点に収束する様子が分かるだろう。

1) TikZ による。