4.4 偶数の分解

それではゴールドバッハの予想を確認してみよう。これまでにプログラムの蓄積があるので、比較的作りやすいはずだ。[TwinPnum.vba] を少し手直ししておこう。[TwinPnum.vba] では $6m \pm 1$ の 2 つの数の素数判定をしたので、その部分を替えるだけで済むだろう。つまり、偶数を 2 つの奇数に分けたとき、それらの素数判定をするように組めばよいだけだ。

programming list [Goldbach.vba]

```
 1 : Sub Goldbach()
 2 : Dim r, m, row As Integer
 3 :
 4 :     m = Sheet1.Cells(1, 1).Value
 5 :
 6 :     r = 3: row = 1
 7 :     While r <= m / 2
 8 :         If pchk(r) And pchk(m - r) = True Then
 9 :             row = row + 1
10 :             Sheet1.Cells(row, 1) = r: Sheet1.Cells(row, 2) = m - r
11 :         End If
12 :         r = r + 2
13 :     Wend
14 : End Sub
15 :
16 : Function pchk(n)
17 : Dim i As Integer
18 : Dim flag As Boolean
19 :
20 :     i = 3: flag = True
21 :     While i <= Sqr(n)
22 :         If n Mod i = 0 Then
23 :             flag = False
24 :             GoTo OUT
25 :         End If
26 :         i = i + 2
27 :     Wend
28 : OUT:
29 :     pchk = flag
30 : End Function
```

プログラムは長ったらしいが新しいことはない。16: 〜 30: 行目までは [TwinPnum.vba] と同じである。SUP をなくしたのは、A1 セルに入力したある偶数に対して、素数の和に直すようなマクロにしたからだ。上限 SUP を決めて、そこまでの偶数の分解の一覧を表示させるには、もうひとつ For-Next を必要とする。それは、君たちの課題にしておきたい。結局のところ、[Goldbach.vba] の本質は 6: 〜 13: 行目である。

まず 4: 行目で、候補となる偶数が A1 セルから読み取られる。

6: 行目の変数 r は、For-Next 構文で m を 2 つの素数に分解するとき、一方の素数を代入するための変数になる。では、もう一方の素数を代入する変数は用意しなくていいの？　そう、その必要はない。なぜなら、もう一方の素数は m - r だからだ。プログラムでは不必要な変数は使わないほうがよろしい。

8: 行目は [TwinPnum.vba] と同様の処理をしている。r と m - r が共に素数かどうか調べているのだ。共に素数のときに限り判定は真となる。調べる範囲が r <= m / 2 であることに注意してほしい。もちろん、調べる範囲が r <= m / 2 で十分な理由は分かるね？

そして見事素数の和に分解されたとき、10: 行目でワークシートに表示される。[TwinPnum.vba] と同様、A 列と B 列にペアで表示している。

[Goldbach.vba] は、入力された偶数に対して、確実に素数の和に分解してくれるだろう。それも、すべてもれなく表示してくれる。ちょっとだけ困るのは、$4 = 2 + 2$ を示すことができない点だ。$4 = 2 + 2$ の表示は自明のことだから構わないといえば構わないのだけれど。

さて、これで 6 以上の偶数の分解がコンピュータ任せにできた。ところが 6 以上の偶数に対しては問題ないのだが、[Goldbach.vba] には致命的な欠陥があるのだ。それは、うっかり奇数を入力してしまった場合に表面化する。奇数 m が入力されると、プログラムはこれを 3 以上の奇数 r と m - r に分けて素数判定に回す。しかし m - r は偶数なのだ。

このときは大変困ったことになってしまう。というのは、関数 pchk(n) は奇数を受け取ることしか想定してないからだ。pchk(n) は受け取った数を、3 から先の奇数で順次割り算を試して、どうにも割れないときに素数と判定する。ところが偶数を受け取った場合、3 から先の奇数で順次割り算を試しても、どうにも割れない数がある。例えば 8 がそうだ。すなわち、偶数が素数と判定されてしまうのだ。

これを回避するには、奇数が入力されたら処理をしなければよい。これは次のように 4: 行目の直後に If 文を加えることで対処できる。ラベル FIN: は 14: 行目の直前に書いておく。

programming list [entrychk.vba]

```
4:      m = Sheet1.Cells(1, 1).Value
*:      If m Mod 2 = 1 Then
*:          GoTo FIN
*:      End If
```

これで奇数入力の危機は一応去る。しかし、マクロは 6 以上の偶数を入力しないと正しく機能しない。2 や 4 が入力されては困る。それを回避するには、もうちょっとコードを書き加えなくてはならない。これは君たちに任せよう。