連分数と連分数展開による関数計算の実際

(1) 連分数とは

連分数（continued fraction）とは，以下のように分母にさらに分数が含まれている分数です。

$$x = a_0 + \cfrac{b_0}{a_1 + \cfrac{b_1}{a_2 + \cfrac{b_2}{a_3}}}$$

連分数のうち，分子がすべて 1（上の例では，$b_0 = b_1 = b_2 = 1$）のものを，特に正則連分数（regular continued fraction）といいます。単に連分数というとき，正則連分数を指すことが多いようです。正則連分数の具体例を以下に示します。

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3}}}$$

(2) 連分数の表記法のバラエティ

連分数表記は，(1)の方法が直感的に分かりやすいのですが，縦方向にどんどん伸びてしまいますので，次のように略記する方法もあります。

$$x = a_0 + \frac{1}{a_1 +}\,\frac{1}{a_2 +}\,\frac{1}{a_3 +}$$

または，正則連分数であることを前提として，

$$x = [a_0; a_1, a_2, a_3]$$

と表記する方法もあります。さらに，級数と同様，無限に続く連分数も考えることができます。

$$x = [a_0; a_1, a_2, a_3, \cdots] = \lim_{n\to\infty}[a_0; a_1, a_2, a_3, \cdots a_n]$$

(3) 連分数の計算方法

ある数 x が与えられたとき，x を超えない最大の整数を a_0 とし，

$$x = a_0 + \frac{1}{x_1}$$

を満足するよう x_1 を設定します。さらに，x_1 が整数でなければ，x_1 を超えない最大の整数を a_1 とし，

$$x_1 = a_1 + \frac{1}{x_2}$$

を満足するよう x_2 を設定します。以下，これを繰り返して，連分数

$$x_1 = a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{x_4}}}}$$

を得ることができます。連分数の数字列（文字列）に変換するプログラム例を以下に示しますので，色々ためしてみましょう。なお，後が乱れますが，誤差のせいですのでご勘弁ください。

【プログラム例】

```
Public Function contFrac(X) As String
  Dim S As String: B = Fix(X)
  If Abs(B - X) < 0.0000001 Then
    contFrac = B
    Exit Function
  End If
  S = B
  For i = 1 To 20
    X = 1 / (X - B): B = Fix(X): S = S & " : " & B
  Next
  contFrac = S
End Function
```

	値	連分数表示
1	1.000000	1
2	1.414214	1 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2 : 2
3	1.732051	1 : 1 : 2 : 1 : 2 : 1 : 2 : 1 : 2 : 1 : 2 : 1 : 2 : 1 : 2 : 1 : 2 : 1 : 2 : 1 : 2 : 1 : 2 : 1 : 2
4	2.000000	2
5	2.236068	2 : 4 : 4 : 4 : 4 : 4 : 4 : 4 : 4 : 4 : 4 : 4 : 4 : 4 : 2 : 1 : 10 : 2 : 1 : 7 : 8 : 1
6	2.449490	2 : 2 : 4 : 2 : 4 : 2 : 4 : 2 : 4 : 2 : 4 : 2 : 4 : 2 : 4 : 2 : 4 : 2 : 4 : 2 : 2 : 1 : 48 : 2 : 4
7	2.645751	2 : 1 : 1 : 1 : 4 : 1 : 1 : 1 : 4 : 1 : 1 : 1 : 4 : 1 : 1 : 1 : 4 : 1 : 1 : 1 : 4
8	2.828427	2 : 1 : 4 : 1 : 4 : 1 : 4 : 1 : 4 : 1 : 4 : 1 : 4 : 1 : 4 : 1 : 4 : 1 : 4 : 1 : 4 : 1 : 5
9	3.000000	3
10	3.162278	3 : 6 : 6 : 6 : 6 : 6 : 6 : 6 : 6 : 6 : 7 : 1 : 1 : 1 : 10 : 5 : 16 : 1 : 16 : 12 : 31
11	3.316625	3 : 3 : 6 : 3 : 6 : 3 : 6 : 3 : 6 : 3 : 6 : 3 : 6 : 6 : 1 : 1 : 2 : 2 : 43 : 2 : 15
12	3.464102	3 : 2 : 6 : 2 : 6 : 2 : 6 : 2 : 6 : 2 : 6 : 2 : 6 : 2 : 5 : 33 : 1 : 3 : 1 : 1 : 3
13	3.605551	3 : 1 : 1 : 1 : 1 : 6 : 1 : 1 : 1 : 1 : 6 : 1 : 1 : 1 : 1 : 6 : 1 : 1 : 1 : 1 : 6
14	3.741657	3 : 1 : 2 : 1 : 6 : 1 : 2 : 1 : 6 : 1 : 2 : 1 : 6 : 1 : 2 : 1 : 6 : 1 : 2 : 1 : 6
15	3.872983	3 : 1 : 6 : 1 : 6 : 1 : 6 : 1 : 6 : 1 : 6 : 1 : 6 : 1 : 6 : 1 : 6 : 1 : 8 : 1 : 1
16	4.000000	4
17	4.123106	4 : 8 : 8 : 8 : 8 : 8 : 8 : 8 : 8 : 7 : 3 : 1 : 1 : 1 : 1 : 1 : 1 : 6 : 2 : 1 : 1
18	4.242641	4 : 4 : 8 : 4 : 8 : 4 : 8 : 4 : 8 : 4 : 7 : 1 : 1 : 8 : 2 : 3 : 1 : 1 : 3 : 4 : 3
19	4.358899	4 : 2 : 1 : 3 : 1 : 2 : 8 : 2 : 1 : 3 : 1 : 2 : 8 : 2 : 1 : 3 : 1 : 2 : 9 : 12 : 8
20	4.472136	4 : 2 : 8 : 2 : 8 : 2 : 8 : 2 : 8 : 2 : 8 : 2 : 8 : 1 : 2 : 5 : 5 : 1 : 3 : 17 : 6
21	4.582576	4 : 1 : 1 : 2 : 1 : 1 : 8 : 1 : 1 : 2 : 1 : 1 : 8 : 1 : 1 : 2 : 1 : 1 : 8 : 1 : 1
22	4.690416	4 : 1 : 2 : 4 : 2 : 1 : 8 : 1 : 2 : 4 : 2 : 1 : 8 : 1 : 2 : 4 : 2 : 1 : 9 : 2 : 6
23	4.795832	4 : 1 : 3 : 1 : 8 : 1 : 3 : 1 : 8 : 1 : 3 : 1 : 8 : 1 : 3 : 1 : 8 : 1 : 3 : 1 : 2
24	4.898979	4 : 1 : 8 : 1 : 8 : 1 : 8 : 1 : 8 : 1 : 8 : 1 : 8 : 1 : 8 : 1 : 5 : 1 : 23 : 1 : 2
25	5.000000	5
26	5.099020	5 : 10 : 10 : 10 : 10 : 10 : 10 : 10 : 7 : 4 : 1 : 6 : 83 : 1 : 12 : 2 : 3 : 11 : 1 : 1 : 1
27	5.196152	5 : 5 : 10 : 5 : 10 : 5 : 10 : 5 : 10 : 4 : 1 : 11 : 3 : 2 : 4 : 8 : 2 : 1 : 1 : 1 : 7
28	5.291503	5 : 3 : 2 : 3 : 10 : 3 : 2 : 3 : 10 : 3 : 2 : 3 : 10 : 2 : 1 : 3 : 3 : 1 : 25 : 3 : 1
29	5.385165	5 : 2 : 1 : 1 : 2 : 10 : 2 : 1 : 1 : 2 : 10 : 2 : 1 : 1 : 2 : 10 : 2 : 1 : 1 : 2 : 1
30	5.477226	5 : 2 : 10 : 2 : 10 : 2 : 10 : 2 : 10 : 2 : 10 : 2 : 10 : 1 : 2 : 4 : 1 : 1 : 11 : 2 : 2
31	5.567764	5 : 1 : 1 : 3 : 5 : 3 : 1 : 1 : 10 : 1 : 1 : 3 : 5 : 3 : 1 : 1 : 10 : 1 : 1 : 7 : 8
32	5.656854	5 : 1 : 1 : 1 : 10 : 1 : 1 : 1 : 10 : 1 : 1 : 1 : 10 : 1 : 1 : 1 : 10 : 1 : 1 : 1 : 11
33	5.744563	5 : 1 : 2 : 1 : 10 : 1 : 2 : 1 : 10 : 1 : 2 : 1 : 10 : 1 : 2 : 1 : 10 : 1 : 2 : 1 : 10
34	5.830952	5 : 1 : 4 : 1 : 10 : 1 : 4 : 1 : 10 : 1 : 4 : 1 : 10 : 1 : 4 : 1 : 10 : 1 : 55 : 1 : 3
35	5.916080	5 : 1 : 10 : 1 : 10 : 1 : 10 : 1 : 10 : 1 : 10 : 1 : 10 : 1 : 11 : 23 : 1 : 5 : 2 : 1 : 1
36	6.000000	6
37	6.082763	6 : 12 : 12 : 12 : 12 : 12 : 12 : 12 : 1 : 1 : 2 : 3 : 1 : 1 : 8 : 3 : 5 : 1 : 19 : 1 : 9
38	6.164414	6 : 6 : 12 : 6 : 12 : 6 : 12 : 6 : 11 : 1 : 3 : 1 : 1 : 1 : 1 : 3 : 1 : 36 : 1 : 2 : 3
39	6.244998	6 : 4 : 12 : 4 : 12 : 4 : 12 : 4 : 12 : 4 : 38 : 3 : 2 : 4 : 1 : 4 : 2 : 9 : 2 : 92 : 10
40	6.324555	6 : 3 : 12 : 3 : 12 : 3 : 12 : 3 : 12 : 3 : 15 : 3 : 5 : 10 : 8 : 2 : 8 : 24 : 15 : 1 : 1
41	6.403124	6 : 2 : 2 : 12 : 2 : 2 : 12 : 2 : 2 : 12 : 2 : 2 : 12 : 2 : 1 : 5 : 1 : 7 : 1 : 1 : 3
42	6.480741	6 : 2 : 12 : 2 : 12 : 2 : 12 : 2 : 12 : 2 : 12 : 2 : 288 : 1 : 5 : 1 : 1 : 3 : 7 : 2 : 2
43	6.557439	6 : 1 : 1 : 3 : 1 : 5 : 1 : 3 : 1 : 1 : 12 : 1 : 1 : 3 : 1 : 5 : 1 : 3 : 1 : 1 : 11
44	6.633250	6 : 1 : 1 : 1 : 2 : 1 : 1 : 1 : 12 : 1 : 1 : 1 : 2 : 1 : 1 : 1 : 12 : 1 : 1 : 1 : 2
45	6.708204	6 : 1 : 2 : 2 : 2 : 1 : 12 : 1 : 2 : 2 : 2 : 1 : 12 : 1 : 2 : 2 : 2 : 1 : 13 : 9 : 3
46	6.782330	6 : 1 : 3 : 1 : 1 : 2 : 6 : 2 : 1 : 1 : 3 : 1 : 12 : 1 : 3 : 1 : 1 : 2 : 6 : 2 : 1
47	6.855655	6 : 1 : 5 : 1 : 12 : 1 : 5 : 1 : 12 : 1 : 5 : 1 : 12 : 1 : 5 : 1 : 11 : 1 : 1 : 1 : 3
48	6.928203	6 : 1 : 12 : 1 : 12 : 1 : 12 : 1 : 12 : 1 : 12 : 1 : 12 : 1 : 10 : 16 : 1 : 8 : 8 : 3 : 5
49	7.000000	7
50	7.071068	7 : 14 : 14 : 14 : 14 : 14 : 14 : 11 : 1 : 1 : 6 : 2 : 1 : 7 : 262 : 1 : 7 : 3 : 4 : 2 : 1
51	7.141428	7 : 7 : 14 : 7 : 14 : 7 : 14 : 7 : 20 : 8 : 39 : 5 : 5 : 7 : 4 : 2 : 6 : 1 : 1 : 1 : 3
52	7.211103	7 : 4 : 1 : 2 : 1 : 4 : 14 : 4 : 1 : 2 : 1 : 4 : 14 : 4 : 1 : 3 : 7 : 1 : 5 : 3 : 11
53	7.280110	7 : 3 : 1 : 1 : 3 : 14 : 3 : 1 : 1 : 3 : 14 : 3 : 1 : 1 : 3 : 14 : 1 : 4 : 2 : 1 : 3
54	7.348469	7 : 2 : 1 : 6 : 1 : 2 : 14 : 2 : 1 : 6 : 1 : 2 : 14 : 2 : 1 : 6 : 1 : 1 : 2 : 1 : 1
55	7.416198	7 : 2 : 2 : 2 : 14 : 2 : 2 : 2 : 14 : 2 : 2 : 2 : 14 : 2 : 2 : 2 : 1 : 1 : 6 : 1 : 22 : 2
56	7.483315	7 : 2 : 14 : 2 : 14 : 2 : 14 : 2 : 14 : 2 : 14 : 2 : 14 : 2 : 2 : 1 : 1 : 15 : 1 : 6 : 2 : 1 : 5
57	7.549834	7 : 1 : 1 : 4 : 1 : 1 : 14 : 1 : 1 : 4 : 1 : 1 : 14 : 1 : 1 : 4 : 1 : 1 : 14 : 1 : 1
58	7.615773	7 : 1 : 1 : 1 : 1 : 1 : 1 : 1 : 14 : 1 : 1 : 1 : 1 : 1 : 1 : 14 : 1 : 1 : 1 : 1 : 1 : 1
59	7.681146	7 : 1 : 2 : 7 : 2 : 1 : 14 : 1 : 2 : 7 : 2 : 1 : 14 : 1 : 2 : 7 : 1 : 1 : 2 : 9 : 11
60	7.745967	7 : 1 : 2 : 1 : 14 : 1 : 2 : 1 : 14 : 1 : 2 : 1 : 14 : 1 : 2 : 1 : 14 : 1 : 4 : 10 : 1
61	7.810250	7 : 1 : 4 : 3 : 1 : 2 : 2 : 1 : 3 : 4 : 1 : 14 : 1 : 4 : 3 : 1 : 2 : 2 : 1 : 4 : 5
62	7.874008	7 : 1 : 6 : 1 : 14 : 1 : 6 : 1 : 14 : 1 : 6 : 1 : 14 : 1 : 6 : 1 : 18 : 4 : 1 : 1 : 1
63	7.937254	7 : 1 : 14 : 1 : 14 : 1 : 14 : 1 : 14 : 1 : 14 : 1 : 15 : 71 : 2 : 1 : 1 : 2 : 3 : 19 : 3
64	8.000000	8
		0
ネイピア数e	2.718282	2 : 1 : 2 : 1 : 1 : 4 : 1 : 1 : 6 : 1 : 1 : 8 : 1 : 1 : 10 : 1 : 1 : 12 : 1 : 1 : 11
π	3.141593	3 : 7 : 15 : 1 : 292 : 1 : 1 : 1 : 2 : 1 : 3 : 1 : 14 : 3 : 3 : 23 : 1 : 1 : 7 : 4 : 35

(4) 連分数の例

A. 2 次方程式の解としての連分数

今，次のような 2 次方程式を考えてみましょう。

$$x^2 - 2x - 1 = 0 \qquad \left(x = 1 \pm \sqrt{2} \right)$$

この式を変形すると，

$$x^2 = 2x + 1 \quad \to x = 2 + \frac{1}{x} \quad \to x = 2 + \frac{1}{2 + \frac{1}{x}} \quad \to x = 2 + \frac{1}{2 + \frac{1}{2 + \frac{1}{x}}} \cdots$$

すなわち，$[2; 2, 2, \cdots]$ は，$x^2 - 2x - 1 = 0$ の解のうち正符号の解であることが分かります。すなわち，

$$x = 1 + \sqrt{2} \quad \therefore \sqrt{2} = [1; 2, 2, 2, \cdots]$$

とすることができます。

B. 黄金数（golden number）φ

連分数の世界では，特に $x^2 - x - 1 = 0$ の解を黄金数といいます。なお，復活祭日を決めるためにキリスト教圏で用いられる数字, すなわち西暦年数に 1 を加えて 19 で割った値も黄金数（golden number）といいますが，これとは別物です。

変形すると，

$$x^2 = x + 1 \quad \to x = 1 + \frac{1}{x} \quad \to x = 1 + \frac{1}{1 + \frac{1}{x}} \quad \to x = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{x}}} \cdots$$

すなわち，黄金数 $\varphi = [1; 1, 1, 1, \cdots]$ となります。

C. 平方根の計算

$x^2 - 2nx - m = 0$ の解が $n + \sqrt{n^2 + m}$ であることは, 解の公式から明らかです。今，連分数を

$$f(n) = \frac{1}{k + \frac{1}{2n + f(n)}}, \quad x = n + f(n)$$

の形で表すと，

$$f(n)\left(k+\frac{1}{2n+f(n)}\right)=1 \quad \rightarrow f(n)\left(\frac{1}{2n+f(n)}\right)=1-kf(n)$$

$$f(n)=(1-kf(n))(2n+f(n))=2n+f(n)-2nkf(n)-kf^2(n)$$

$$\therefore kf^2(n)+2nkf(n)=2n \quad \rightarrow f^2(n)+2nf(n)=\frac{2n}{k}$$

一方，　$\sqrt{n^2+m}=n+f(n) \quad \rightarrow n^2+m=n^2+2nf(n)+f^2(n)$

ですから　$m=f^2(n)+2nf(n)$　となります。したがって

$$k=\frac{2n}{m}$$

を得ることができます。たとえば，$n=1, m=2$ のとき，すなわち

$$\sqrt{3}=[1;1,2,1,2,\cdots]$$

となるのはこのためです。

本来の定義では,連分数の各係数は整数ですが,整数にこだわらなければ,このことを使って平方根を求めることができます。すなわち,

$$\sqrt{n^2+m}=\left[\ n;\frac{2n}{m},2n,\frac{2n}{m},2n,\frac{2n}{m},2n,\cdots\right]$$

とすることで平方根を求めることができます。以下にそのプログラミング例を VB で示します。

【プログラム例】

```
Public Function mySqr(X)
   XX = Abs(X)
   If XX < 0.00000000001 Then
       mySqr = 0: Exit Function
   End If
   less1 = False    '1より小さい時，逆数をとる
   If XX < 1 Then
     XX = 1 / XX: less1 = True
   End If
   N = 1: NN = 1  '平方がXXを超えない整数を求める
   Do While (NN < XX)
     N = N + 1: NN = N * N
```

```
    Loop
    If Abs(NN - XX) < 0.00000000001 Then
        mySqr = N: Exit Function
    End If
    N = N - 1: NN = N * N
    DX = XX - NN
    If Abs(DX) < 0.00000000001 Then
        mySqr = N: Exit Function
    End If
    DN = 2 * N: DM = DN / DX    '連分数の計算
    Dim Z As Double: Z = 0
    For i = 1 To 20
       Z = 1 / (DM + 1 / (DN + Z))
    Next
    mySqr = N + Z    '元データが1より小さい時，逆数が結果
    If less1 Then mySqr = 1 / mySqr
End Function
```

D. 白銀数

$1+\sqrt{2}=[2;2,2,2,2,\cdots]$ を白銀数といいます。その逆数は,

$$\frac{1}{1+\sqrt{2}}=\frac{1-\sqrt{2}}{1-2}=-1+\sqrt{2}=[0;2,2,2,2,\cdots]$$

となります。

E. 円周率πの値

円周率の正則連分数には,以下のように規則性がないと考えられています。

$$\pi=[3;7,15,1,292,1,1,1,2,1,3,\cdots]$$

ただ，正則でない連分数では規則性を持つ場合があります。

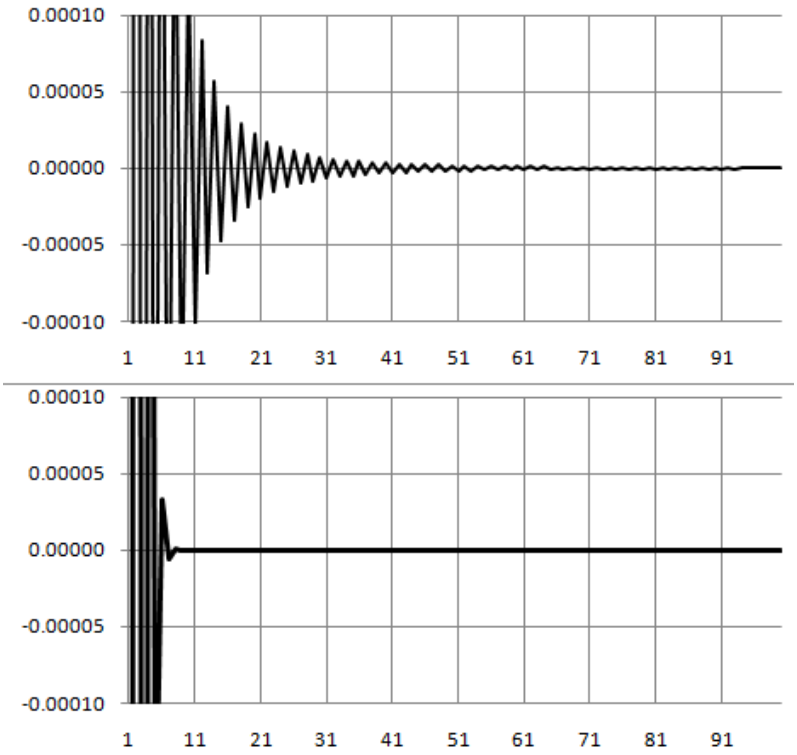
$$\pi=3+\cfrac{1^2}{6+\cfrac{3^2}{6+\cfrac{5^2}{\ddots}}} \qquad \frac{4}{\pi}=1+\cfrac{1^2}{3+\cfrac{2^2}{5+\cfrac{3^2}{\ddots}}}$$

円周率を計算するには，右のほうが早く収束するようです。たとえば，Excel VBA で次のような関数を作成して確認してみましょう。引数の N は繰返し回数です。N を変えて比較すると収束の様子が分かります。

【左の連分数】
```
Public Function myPi1(N)
  NN = N * 2 + 1: A = 7
  For i = NN To 1 Step -2
    A = 6 + (i * i) / A
  Next
  myPi1 = A - 3
End Function
```

【右の連分数】
```
Public Function myPi2(N)'
  A = 2 * N
  For i = N To 1 Step -1
    A = 2 * i - 1 + (i * i) / A
  Next
  myPi2 = 4 / A
End Function
```

収束の様子をグラフ化してみました。上図が左の連分数，下図が右の連分数です。



左の連分数

右の連分数

F. ネイピア数 e（自然対数の底）

ネイピア数の連分数展開は循環しませんが，規則性を持っています。すなわち，以下のようになります。

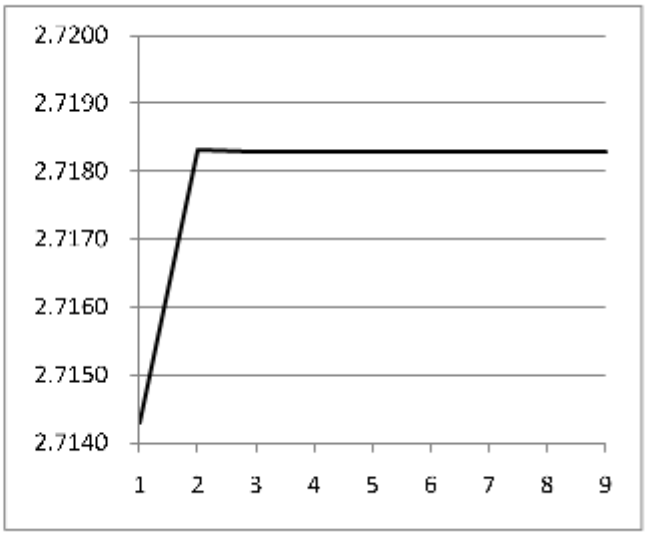
$$e = [\,2; 1, 2, 1, 1, 4, 1, 1, 8, 1, 1, 10, 1, 1, 12, 1, 1, 14, \cdots]$$

整数部以外は，$1, 2n, 1$ $(n = 1, 2, \cdots, \infty)$ が続いています。これを利用してネイピア数を計算することができます。

【プログラム例】

```
Public Function myExp(N) As Double
  Dim A As Double
  NN = N * 2: A = 1
  For i = NN To 2 Step -2
    A = 1 + 1 / (i + 1 / (1 + 1 / A))
  Next
  myExp = 2 + 1 / A
End Function
```

繰返回数	ネイピア数
1	2.71428571428571000
2	2.71830985915493000
3	2.71828171828172000
4	2.71828182873570000
5	2.71828182845856000
6	2.71828182845905000
7	2.71828182845905000
8	2.71828182845905000
9	2.71828182845905000
10	2.71828182845905000
11	2.71828182845905000
12	2.71828182845905000
13	2.71828182845905000
14	2.71828182845905000
15	2.71828182845905000
16	2.71828182845905000



(5)連分数の性質

A. 数列 p_n, q_n の定義

いま，正の整数 $a_n (n = 0, 1, 2, 3, \cdots)$ の数列，$a_0, a_1, a_2, a_3, \cdots$ を考え，数列 p_n, q_n を

$$p_0 = 1, \quad p_1 = a_0, \quad p_n = a_{n-1} \cdot p_{n-1} + p_{n-2} \quad (n \geq 2)$$

$$q_0 = 0, \quad q_1 = 1, \quad q_n = a_{n-1} \cdot q_{n-1} + q_{n-2} \quad (n \geq 2)$$

とすると，連分数は

$$[a_0; a_1, a_2, \cdots, a_{n-1}] = \frac{a_{n-1}p_{n-1} + p_{n-2}}{a_{n-1}q_{n-1} + q_{n-2}} = \frac{p_n}{q_n}$$

となることが知られています。

B. ユークリッドの互除法との関係

数列 p_n の定義を逆に読むと「被除数 p_n を除数 p_{n-1} で除した商を a_{n-1},とし，新しい被除数を p_{n-1} とし，除数を余り p_{n-2} とする」と読むことができます。

すなわち，n の方向を逆に辿ればユークリッドの互除法を計算し，その過程で現れた商が連分数の数列となっているわけです。いわば，上記数列の定義は互除法の操作を逆に辿っていると考えることができます。

さらに，p_n, q_n は整数であり，かつユークリッドの互除法を適用していますので，p_n と q_n は互いに素の関係にあります。すなわち，p_n/q_n は，すでに通分された関係ですから，既約分数となります。

C. p_n と q_n の関係

次のような計算を行ってみましょう。

$$\begin{aligned} p_1q_0 - p_0q_1 &= 1\times 0 - 1\times 1 = -1 \\ p_2q_1 - p_1q_2 &= (a_1p_1 + p_0)q_1 - p_1(a_1q_1 + q_0) \\ &= -(p_1q_0 - p_0q_1) = 1 \end{aligned}$$

いま，$p_nq_{n-1} - p_{n-1}q_n = \pm 1$ が成り立つとすれば，

$$\begin{aligned} p_{n+1}q_n - p_nq_{n+1} &= (a_np_n + p_{n-1})q_n - p_n(a_nq_n + q_{n-1}) \\ &= a_np_nq_n + p_{n-1}q_n - a_np_nq_n - p_nq_{n-1} \\ &= -(p_nq_{n-1} - p_{n-1}q_n) = \mp 1 \end{aligned}$$

が成り立ちますので，数学的帰納法により

$$p_nq_{n-1} - p_{n-1}q_n = (-1)^n$$

が成立します。

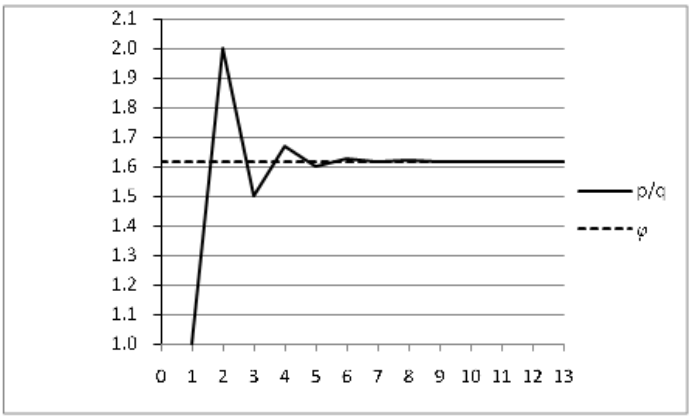
D. 黄金数のとき

数列 a_n がすべて 1 のとき，数列 p_n, q_n はフィボナッチ数列 $(F_0 = 0, F_1 = 1)$ の形をしています。ですから，

$$\frac{p_n}{q_n} = \frac{F_{n+1}}{F_n}$$

の関係が成立します。このとき，連分数は黄金数ですから，隣り合うフィボナッチ数列の比は黄金数に収束することが分かります。この様子を Excel の計算式として定義し，グラフを描いたものを以下に示します。以下のように黄金数に収束している様子が分かります。

n	a_n	p	q	p/q	φ
0	1	1	0		1.61803
1	1	1	1	1.00000	1.61803
2	1	2	1	2.00000	1.61803
3	1	3	2	1.50000	1.61803
4	1	5	3	1.66667	1.61803
5	1	8	5	1.60000	1.61803
6	1	13	8	1.62500	1.61803
7	1	21	13	1.61538	1.61803
8	1	34	21	1.61905	1.61803
9	1	55	34	1.61765	1.61803
10	1	89	55	1.61818	1.61803
11	1	144	89	1.61798	1.61803
12	1	233	144	1.61806	1.61803
13	1	377	233	1.61803	1.61803
14	1	610	377	1.61804	1.61803
15	1	987	610	1.61803	1.61803



(6) 連分数を使った関数計算

A. 非常に収束の遅い級数の場合

関数のマクローリン展開では，以下のように，非常に収束が遅い級数もあります。

$$\log_e x = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots$$

このようなとき，前述した平方根の計算のように，関数を連分数の形で展開して計算することで，繰返し回数が極端に減る場合があります。以下，それぞれの連分数展開の形と例題としてのプログラムを示します。

B. 対数（log）

【連分数】 $\log_e x = x/(1 + x/(2 + x/(3 + 2x/(2 + 2x/(5 + \cdots$

$$+ nx/(2 + nx/((2n + 1) + \cdots$$

【計算上の補足】 あらかじめ $1 \le x < 2$ となるよう調整する必要があります。このため，何らかの値で除算を繰り返すことで，この範囲に収めます。たとえば，元の値が $2^n x\ (1 \le x < 2)$ の場合，次のように考えて計算します。

$$\log_e 2^n x = \log_e 2^n + \log_e x = n \log_e 2 + \log_e x$$

この場合，値 2 による除算を繰り返し，その回数分だけ $\log_e 2$ を加算します。次のように考えても構いません。

$$\log_e e^n x = \log_e e^n + \log_e x = n + \log_e x$$

この場合，値 e による除算を繰り返し，その回数分だけ 1 を加算します。

ただし，値 e 自身が近似値を使わざるをえないこと，さらに値 2 による除算の方が精度が良い等の理由で，前者の方法が若干良いようです。

【プログラム例１】

```
Public Function myLog(X)
 log2 = 0.693147180559945: XX = Abs(X)
 If XX < 0.0000000000001 Then
   myLog = -1E+308
   Exit Function
 End If
 L = 0
 Do While XX >= 2
   XX = XX / 2: L = L + log2
 Loop
 Do While XX < 1
   XX = XX * 2: L = L - log2
 Loop
 XX = XX - 1: A = 0: N = 20
 For i = N To 1 Step -1
   A = i * XX / (2 + i * XX / (2 * i + 1 + A))
 Next
 myLog = L + XX / (1 + A)
End Function
```

【プログラム例 2】

```
Public Function myLog2(X)
 expC = 2.71828182845905: XX = Abs(X)
 If XX < 0.0000000000001 Then
   myLog2 = -1E+308
   Exit Function
 End If
 L = 0
 Do While XX >= 2 * expC
   XX = XX / expC: L = L + 1
 Loop
 Do While XX < 1
   XX = XX * expC: L = L - 1
 Loop
 XX = XX - 1: A = 0: N = 20
 For i = N To 1 Step -1
   A = i * XX / (2 + i * XX / (2 * i + 1 + A))
 Next
 myLog2 = L + XX / (1 + A)
End Function
```


以下は，Excel に備わっている ln 関数と連分数展開のプログラムとの比較です。ほぼ同じ値であることが分かります。

x	log	myLog(x)	myLog(x)-log(x)	mylog2(x)	myLog2(X)-log(x)
0.1	-2.3025850929940500	-2.3025850929940400	0.0000000000000000	-2.3025850929940400	0.0000000000000488
0.2	-1.6094379124341000	-1.6094379124341000	0.0000000000000000	-1.6094379124341000	0.0000000000000333
0.3	-1.2039728043259400	-1.2039728043259400	0.0000000000000000	-1.2039728043259300	0.0000000000000355
0.4	-0.9162907318741550	-0.9162907318741540	0.0000000000000000	-0.9162907318741530	0.0000000000000167
0.5	-0.6931471805599450	-0.6931471805599450	0.0000000000000000	-0.6931471805599440	0.0000000000000167
0.6	-0.5108256237659910	-0.5108256237659900	0.0000000000000000	-0.5108256237659890	0.0000000000000178
0.7	-0.3566749439387320	-0.3566749439387320	0.0000000000000000	-0.3566749439387310	0.0000000000000172
0.8	-0.2231435513142100	-0.2231435513142090	0.0000000000000033	-0.2231435513142080	0.0000000000000167
0.9	-0.1053605156578260	-0.1053605156578260	0.0000000000000028	-0.1053605156578250	0.0000000000000172
1.0	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
1.1	0.0953101798043249	0.0953101798043249	0.0000000000000000	0.0953101798043249	0.0000000000000000
1.2	0.1823215567939550	0.1823215567939550	0.0000000000000000	0.1823215567939550	0.0000000000000000
1.3	0.2623642644674910	0.2623642644674910	0.0000000000000000	0.2623642644674910	0.0000000000000000
1.4	0.3364722366212130	0.3364722366212130	0.0000000000000000	0.3364722366212130	0.0000000000000000
1.5	0.4054651081081650	0.4054651081081650	0.0000000000000000	0.4054651081081650	0.0000000000000000
1.6	0.4700036292457360	0.4700036292457360	0.0000000000000000	0.4700036292457360	0.0000000000000000
1.7	0.5306282510621700	0.5306282510621700	0.0000000000000000	0.5306282510621700	0.0000000000000000
1.8	0.5877866649021190	0.5877866649021190	0.0000000000000000	0.5877866649021190	0.0000000000000000
1.9	0.6418538861723950	0.6418538861723950	0.0000000000000000	0.6418538861723950	0.0000000000000000
2.0	0.6931471805599450	0.6931471805599450	0.0000000000000000	0.6931471805599450	0.0000000000000000
2.1	0.7419373447293770	0.7419373447293770	0.0000000000000000	0.7419373447293770	0.0000000000000000
2.2	0.7884573603642700	0.7884573603642700	0.0000000000000000	0.7884573603642700	0.0000000000000000
2.3	0.8329091229351040	0.8329091229351040	0.0000000000000000	0.8329091229351040	0.0000000000000000
2.4	0.8754687373539000	0.8754687373539000	0.0000000000000000	0.8754687373539000	0.0000000000000000
2.5	0.9162907318741550	0.9162907318741550	0.0000000000000000	0.9162907318741550	0.0000000000000000
2.6	0.9555114450274360	0.9555114450274360	0.0000000000000000	0.9555114450274360	0.0000000000000000
2.7	0.9932517730102830	0.9932517730102830	0.0000000000000000	0.9932517730102840	0.0000000000000000
2.8	1.0296194171811600	1.0296194171811600	0.0000000000000000	1.0296194171811600	0.0000000000000000
2.9	1.0647107369924300	1.0647107369924300	0.0000000000000000	1.0647107369924300	0.0000000000000000
3.0	1.0986122886681100	1.0986122886681100	0.0000000000000000	1.0986122886681100	0.0000000000000000

値 e に近似値を使わざるをえないことから誤差が出ると前述しましたが，倍精度では以下の桁数が限度です。

e = 2.7182 81828 45905

一方，正確な値は e = 2.7182 81828 45904 52353 … です。

そこで，e_1= 2.7182 81828 45904，e_2= 2.7182 81828 45905，α = 0.52353 … として，次のように計算します。

$$p = \frac{\alpha}{1-\alpha},\quad e = \frac{e_1 + pe_2}{1+p}$$

なお，α が 1 に近いときは，次のように計算すれば構いません。

$$p = \frac{1-\alpha}{\alpha},\quad e = \frac{pe_1 + e_2}{1+p}$$

プログラム例を以下に示します。ただし，計算機上での演算は本質的に切り捨てです。除算結果が 0 でないとき，最終桁が切り捨てられますので，この分を最後に加算しておきます。

【プログラム例 3】

```
Public Function myLog3(X)
 expc1 = 2.71828182845904: expc2 = 2.71828182845905
  Alf = 0.523536028747135: P = Alf / (1 - Alf): PP = 1 + P
  XX = Abs(X)
  If XX < 0.0000000000001 Then
    myLog3 = -1E+308
    Exit Function
  End If
  L = 0
  Do While (XX - expc2) >= expc1
    XX = XX * PP / (expc1 + P * excp2): L = L + 1
  Loop
  Do While XX < 1
    XX = XX * (expc1 + P * expc2) / PP: L = L - 1
  Loop
  XX = XX - 1: A = 0: N = 20
  For i = N To 1 Step -1
    A = i * XX / (2 + i * XX / (2 * i + 1 + A))
  Next
  myLog3 = L + XX / (1 + A)
  If myLog3 <> 0 Then myLog3 = myLog3 + 1E-16
 End Function
```

以下のように誤差が非常に少なくなっていることが分かります。

x	log	myLog3(x)	myLog3(x)-log(x)
0.1	-2.3025850929940500	-2.3025850929940500	0.0000000000000000
0.2	-1.6094379124341000	-1.6094379124341000	0.0000000000000000
0.3	-1.2039728043259400	-1.2039728043259400	0.0000000000000000
0.4	-0.9162907318741550	-0.9162907318741550	0.0000000000000000
0.5	-0.6931471805599450	-0.6931471805599450	0.0000000000000000
0.6	-0.5108256237659910	-0.5108256237659910	0.0000000000000000
0.7	-0.3566749439387320	-0.3566749439387320	0.0000000000000000
0.8	-0.2231435513142100	-0.2231435513142100	0.0000000000000000
0.9	-0.1053605156578260	-0.1053605156578260	0.0000000000000000
1.0	0.0000000000000000	0.0000000000000000	0.0000000000000000
1.1	0.0953101798043249	0.0953101798043250	0.0000000000000000
1.2	0.1823215567939550	0.1823215567939550	0.0000000000000000
1.3	0.2623642644674910	0.2623642644674910	0.0000000000000000
1.4	0.3364722366212130	0.3364722366212130	0.0000000000000000
1.5	0.4054651081081650	0.4054651081081650	0.0000000000000000
1.6	0.4700036292457360	0.4700036292457360	0.0000000000000000
1.7	0.5306282510621700	0.5306282510621710	0.0000000000000000
1.8	0.5877866649021190	0.5877866649021190	0.0000000000000000
1.9	0.6418538861723950	0.6418538861723950	0.0000000000000000
2.0	0.6931471805599450	0.6931471805599450	0.0000000000000000
2.1	0.7419373447293770	0.7419373447293780	0.0000000000000000
2.2	0.7884573603642700	0.7884573603642700	0.0000000000000000
2.3	0.8329091229351040	0.8329091229351040	0.0000000000000000
2.4	0.8754687373539000	0.8754687373539000	0.0000000000000000
2.5	0.9162907318741550	0.9162907318741550	0.0000000000000000
2.6	0.9555114450274360	0.9555114450274370	0.0000000000000000
2.7	0.9932517730102830	0.9932517730102840	0.0000000000000000
2.8	1.0296194171811600	1.0296194171811600	0.0000000000000000
2.9	1.0647107369924300	1.0647107369924300	0.0000000000000000
3.0	1.0986122886681100	1.0986122886681100	0.0000000000000000

上記【プログラム例 1】では，log 0.8 および log 0.9 の値の誤差が大きくなっています。これは，L = L - log 2 の部分の桁落ちのためです。log 2 の値をより大きい桁数で示すと以下のようになりますが

log 2=0.6931 47180 55994 53094 17232 …

この値が 15 桁で切られていますので，この部分を補正します。

【プログラム例 4】

```
Public Function myLog3(X)
  N = 20: log2 = 0.693147180559945: Flog2 = 3.09417232E-16
  XX = Abs(X)
  If XX < 0.00000000000001 Then
    myLog4 = -1E+308
    Exit Function
  End If
  L = 0
  Do While XX >= 2
    XX = XX / 2: L = L + log2 + Flog2
  Loop
  Do While XX < 1
    XX = XX * 2: L = L - log2 - Flog2
  Loop
  XX = XX - 1: A = 0
  For i = N To 1 Step -1
    A = i * XX / (2 + i * XX / (2 * i + 1 + A))
  Next
  myLog4 = L + XX / (1 + A)
 End Function
```

x	log	myLog4(x)	myLog4(x)-log(x)
0.1	-2.3025850929940500	-2.3025850929940500	0.0000000000000000
0.2	-1.6094379124341000	-1.6094379124341000	0.0000000000000000
0.3	-1.2039728043259400	-1.2039728043259400	0.0000000000000000
0.4	-0.9162907318741550	-0.9162907318741550	0.0000000000000000
0.5	-0.6931471805599450	-0.6931471805599450	0.0000000000000000
0.6	-0.5108256237659910	-0.5108256237659910	0.0000000000000000
0.7	-0.3566749439387320	-0.3566749439387320	0.0000000000000000
0.8	-0.2231435513142100	-0.2231435513142100	0.0000000000000000
0.9	-0.1053605156578260	-0.1053605156578260	0.0000000000000000
1.0	0.0000000000000000	0.0000000000000000	0.0000000000000000
1.1	0.0953101798043249	0.0953101798043249	0.0000000000000000
1.2	0.1823215567939550	0.1823215567939550	0.0000000000000000
1.3	0.2623642644674910	0.2623642644674910	0.0000000000000000
1.4	0.3364722366212130	0.3364722366212130	0.0000000000000000
1.5	0.4054651081081650	0.4054651081081650	0.0000000000000000
1.6	0.4700036292457360	0.4700036292457360	0.0000000000000000
1.7	0.5306282510621700	0.5306282510621700	0.0000000000000000
1.8	0.5877866649021190	0.5877866649021190	0.0000000000000000
1.9	0.6418538861723950	0.6418538861723950	0.0000000000000000
2.0	0.6931471805599450	0.6931471805599450	0.0000000000000000
2.1	0.7419373447293770	0.7419373447293770	0.0000000000000000
2.2	0.7884573603642700	0.7884573603642700	0.0000000000000000
2.3	0.8329091229351040	0.8329091229351040	0.0000000000000000
2.4	0.8754687373539000	0.8754687373539000	0.0000000000000000
2.5	0.9162907318741550	0.9162907318741550	0.0000000000000000
2.6	0.9555114450274360	0.9555114450274360	0.0000000000000000
2.7	0.9932517730102830	0.9932517730102830	0.0000000000000000
2.8	1.0296194171811600	1.0296194171811600	0.0000000000000000
2.9	1.0647107369924300	1.0647107369924300	0.0000000000000000
3.0	1.0986122886681100	1.0986122886681100	0.0000000000000000

C. 逆正接（arctan）

【連分数】 $\mathrm{Tan}^{-1}x = x/(1+x^2/(3+4x^2/(5+\cdots+n^2x^2/((2n+1)+\cdots$

【計算上の補足】　まず，$x=\tan(A+\pi/2)$ の計算を考えてみると，

$$\mathrm{Tan}^{-1}x = A+\pi/2 \qquad ①$$

一方，$x=\tan(A+\pi/2)=\tan(A+\pi/4+\pi/4)$

$$=\frac{\tan(A+\pi/4)+1}{1-\tan(A+\pi/4)}=\frac{\dfrac{\tan A+1}{1-\tan A}+1}{1-\dfrac{\tan A+1}{1-\tan A}}=\frac{\dfrac{\tan A+1+1-\tan A}{1-\tan A}}{\dfrac{1-\tan A-\tan A-1}{1-\tan A}}$$

$$=-\frac{1}{\tan A}$$

したがって，　$\tan A=-1/x \qquad \rightarrow A=-\mathrm{Tan}^{-1}(1/x) \qquad ②$

上記①式との比較から，

$$\mathrm{Tan}^{-1}(x)=\frac{\pi}{2}-\mathrm{Tan}^{-1}\left(\frac{1}{x}\right)$$

すなわち $x>1$ のとき，$1/x$ として関数値を求め，上記計算を行います。

一方，$x=\tan(A-\pi/2)$ のとき，同様にして，

$$\mathrm{Tan}^{-1}(x)=-\frac{\pi}{2}-\mathrm{Tan}^{-1}\left(\frac{1}{x}\right)$$

が求まりますので，$x<1$ のとき，$1/x$ として関数値を求め，上記計算を行います。

以上をまとめると，$-1\le x\le 1$ でない場合，$\mathrm{T}=\mathrm{Tan}^{-1}(1/x)$ を求め，

$$Tan^{-1}(x)=\begin{cases}\dfrac{\pi}{2}-T & (x>1)\\ -\dfrac{\pi}{2}-T & (x<-1)\end{cases}$$

とする必要があります。

なお，絶対値 $|x|$ に対して上記の計算を行い，最後に符号を調整すると判定が少なくなります。プログラム例では，この方法を用いています。

【プログラム例 1】

```
Public Function myAtan(X)
  N = 20: Pi2 = 3.14159265358979 / 2: XX = X: XX = Abs(X)
  If XX <= 1 Then T = XX Else T = 1 / XX
  A = 0
  For i = N To 1 Step -1
   A = (i * T) ^ 2 / (2 * i + 1 + A)
  Next
  TT = T / (1 + A)
  If XX > 1 Then
     R = Pi2 - TT
  Else
     R = TT
  End If
  If X < 0 Then R = -R
  myAtan = R
End Function
```

以下に Excel の Atan との比較を示します。

x	ExcelのATAN(x)	myAtan(x)	ATAN(x)-myAtan(x)
-2.0	-1.1071487177940900	-1.1071487177940900	0.0000000000000000
-1.8	-1.0636978224025600	-1.0636978224025600	-0.0000000000000018
-1.6	-1.0121970114513300	-1.0121970114513300	0.0000000000000000
-1.4	-0.9505468408120750	-0.9505468408120740	-0.0000000000000016
-1.2	-0.8760580505981930	-0.8760580505981920	-0.0000000000000017
-1.0	-0.7853981633974480	-0.7853981633974480	0.0000000000000000
-0.8	-0.6747409422235530	-0.6747409422235530	0.0000000000000000
-0.6	-0.5404195002705840	-0.5404195002705840	0.0000000000000000
-0.4	-0.3805063771123650	-0.3805063771123650	0.0000000000000000
-0.2	-0.1973955598498810	-0.1973955598498810	0.0000000000000000
0.0	0.0000000000000000	0.0000000000000000	0.0000000000000000
0.2	0.1973955598498810	0.1973955598498810	0.0000000000000000
0.4	0.3805063771123650	0.3805063771123650	0.0000000000000000
0.6	0.5404195002705840	0.5404195002705840	0.0000000000000000
0.8	0.6747409422235530	0.6747409422235530	0.0000000000000000
1.0	0.7853981633974480	0.7853981633974480	0.0000000000000000
1.2	0.8760580505981940	0.8760580505981920	0.0000000000000017
1.4	0.9505468408120750	0.9505468408120740	0.0000000000000017
1.6	1.0121970114513300	1.0121970114513300	0.0000000000000000
1.8	1.0636978224025600	1.0636978224025600	0.0000000000000000
2.0	1.1071487177940900	1.1071487177940900	0.0000000000000000

上記【プログラム例 1】では，以下の部分で桁落ちが生じます。

```
R = Pi2 - TT
```

Pi2 が近似値であるため近似値として記述した桁数より以下の少数桁の分

だけ，R の値が小さくなっています。そこで，次のように Pi2 の近似値以下の分を加えておきます。

```
H = 3.23846264338328 * 0.5: HD = 0.000000000000001 '1E-15
    .
    .
    .
R = Pi2 - TT + H * HD
```

【プログラム例 2】

```
Public Function myAtan2(X)
  Pi2 = 3.14159265358979 * 0.5
  H = 3.23846264338328 * 0.5: HD = 0.000000000000001 '1E-15
  XX = X: XX = Abs(X)
  If XX <= 1 Then T = XX Else T = 1 / XX
  A = 0: N = 30
  For i = N To 1 Step -1
   A = (i * T) ^ 2 / (2 * i + 1 + A)
  Next
  TT = T / (1 + A)
  If XX > 1 Then
     R = Pi2 - TT + H * HD
  Else
     R = TT
  End If
  If X < 0 Then R = -R
  myAtan2 = R
End Function
```

x	ExcelのATAN(x)	myAtan2(x)	ATAN(x)-myAtan2(x)
-2.0	-1.1071487177940900	-1.1071487177940900	0.0000000000000000
-1.8	-1.0636978224025600	-1.0636978224025600	0.0000000000000000
-1.6	-1.0121970114513300	-1.0121970114513300	0.0000000000000000
-1.4	-0.9505468408120750	-0.9505468408120750	0.0000000000000000
-1.2	-0.8760580505981930	-0.8760580505981930	0.0000000000000000
-1.0	-0.7853981633974480	-0.7853981633974480	0.0000000000000000
-0.8	-0.6747409422235530	-0.6747409422235530	0.0000000000000000
-0.6	-0.5404195002705840	-0.5404195002705840	0.0000000000000000
-0.4	-0.3805063771123650	-0.3805063771123650	0.0000000000000000
-0.2	-0.1973955598498810	-0.1973955598498810	0.0000000000000000
0.0	0.0000000000000000	0.0000000000000000	0.0000000000000000
0.2	0.1973955598498810	0.1973955598498810	0.0000000000000000
0.4	0.3805063771123650	0.3805063771123650	0.0000000000000000
0.6	0.5404195002705840	0.5404195002705840	0.0000000000000000
0.8	0.6747409422235530	0.6747409422235530	0.0000000000000000
1.0	0.7853981633974480	0.7853981633974480	0.0000000000000000
1.2	0.8760580505981940	0.8760580505981940	0.0000000000000000
1.4	0.9505468408120750	0.9505468408120750	0.0000000000000000
1.6	1.0121970114513300	1.0121970114513300	0.0000000000000000
1.8	1.0636978224025600	1.0636978224025600	0.0000000000000000
2.0	1.1071487177940900	1.1071487177940900	0.0000000000000000

D. 正接（tan）

【連分数】 $\tan x = x/(1 - x^2/(3 - x^2/(5 - \cdots - x^2/((2n+1) - \cdots$

【計算上の補足】　特に連分数展開でなくても，マクローリン展開，または正弦（sin）および余弦（cos）から求めることができますが，ここではあえて連分数展開で計算する例を示します。

　計算する前に，あらかじめ $-\pi/2 \le x < \pi/2$ となるよう $n\pi$ 分を加算または減算します。たとえば，$\pi/2$ より大きい場合，$n = (x + \pi/2)/\pi$ とした後，整数化して $n\pi$ を差し引きます。

【プログラム例１】

```
Public Function myTan(X)
 N = 10: Pi = 3.14159265358979: HPi = Pi / 2
 If X > 0 Then
    K = Fix((X + HPi) / Pi): XX = X - Pi * K
 Else
    K = Fix((-X - HPi) / Pi): XX = X + Pi * K
 End If
 AX2 = XX * XX: A = 0
 For i = N To 1 Step -1
   A = AX2 / ((i * 2 + 1) - A)
 Next
 myTan = Abs(XX) / (1 - A)
 If XX < 0 Then myTan = -myTan
End Function
```

　実は【プログラム例 1】には重大な欠陥があります。正の場合，

```
XX = X - Pi * K
```

において，桁落ちが起きて有効桁数が少なくなります。Pi が近似値であるため近似値として記述した桁数より以下の少数桁の分だけ，XX の値が大きくなっています。そこで，次のように Pi の近似値以下の分を差し引きます。

```
XX = X - Pi * K: XX = XX - K * 3.2385… * 1E-15
```

　負の場合，

```
XX = X + Pi * K
```

の部分で情報落ちが発生します。ただし，倍精度浮動小数点は 15 桁ですので，これ以上どうしようもありません。一方，

$$\tan(-x) = -\tan x$$

ですから，正の x について計算できれば構いません。そこで，次のように書き換えることにします。

【プログラム例 2】

```
Public Function myTan2(X)
  N = 10: Pi = 3.14159265358979: Hpi = Pi / 2: RPi = 1 / Pi
  H = 3.23846264338328: DH = 0.000000000000001 ' 1E-15
  XX = Abs(X): K = Fix((XX + Hpi) * RPi)
  XX = XX - Pi * K: XX = XX - K * H * DH
  AX2 = XX * XX
  A = 0
  For i = N To 1 Step -1
    A = AX2 / ((i * 2 + 1) - A)
  Next
  myTan2 = Abs(XX) / (1 - A)
  If XX < 0 Then myTan2 = -myTan2
  If X < 0 Then myTan2 = -myTan2
End Function
```

x	ExcelのTAN(x)	myTan(x)	ExcelのTAN(x)-myTan(x)	myTan2(x)	ExcelのTAN(x)-myTan2(x)
-3.1416	0.000000000000000	0.000000000040404	-0.000000000040404	0.000000000000000	0.000000000000000
-2.9845	0.158384440324536	0.158384440337826	-0.000000000013289	0.158384440324536	0.000000000000000
-2.8274	0.324919696232906	0.324919696237211	-0.000000000004305	0.324919696232906	0.000000000000000
-2.6704	0.509525449494429	0.509525449495798	-0.000000000001369	0.509525449494429	0.000000000000000
-2.5133	0.726542528005361	0.726542528005788	-0.000000000000427	0.726542528005361	0.000000000000000
-2.3562	1.000000000000000	1.000000000000130	-0.000000000000131	1.000000000000000	0.000000000000000
-2.1991	1.376381920471170	1.376381920471210	-0.000000000000040	1.376381920471170	0.000000000000000
-2.0420	1.962610505505150	1.962610505505160	-0.000000000000013	1.962610505505150	0.000000000000000
-1.8850	3.077683537175250	3.077683537175260	-0.000000000000004	3.077683537175260	0.000000000000000
-1.7279	6.313751514675050	6.313751514675050	0.000000000000000	6.313751514675050	0.000000000000000
-1.5708					
-1.4137	-6.313751514675040	-6.313746324418690	-0.000005190256350	-6.313751514675040	0.000000000000000
-1.2566	-3.077683537175250	-3.077682904265020	-0.000000632910237	-3.077683537175250	0.000000000000000
-1.0996	-1.962610505505150	-1.962610370345670	-0.000000135159482	-1.962610505505150	0.000000000000000
-0.9425	-1.376381920471170	-1.376381884548840	-0.000000035922337	-1.376381920471170	0.000000000000000
-0.7854	-1.000000000000000	-0.999999989337550	-0.000000010662450	-1.000000000000000	0.000000000000000
-0.6283	-0.726542528005361	-0.726542524641003	-0.000000003364358	-0.726542528005361	0.000000000000000
-0.4712	-0.509525449494429	-0.509525448396286	-0.000000001098143	-0.509525449494429	0.000000000000000
-0.3142	-0.324919696232906	-0.324919695868413	-0.000000000364494	-0.324919696232906	0.000000000000000
-0.1571	-0.158384440324536	-0.158384440202952	-0.000000000121584	-0.158384440324536	0.000000000000000
0.0000	0.000000000000000	0.000000000040407	-0.000000000040407	0.000000000000000	0.000000000000000
0.1571	0.158384440324536	0.158384440324536	0.000000000000000	0.158384440324536	0.000000000000000
0.3142	0.324919696232906	0.324919696232906	0.000000000000000	0.324919696232906	0.000000000000000
0.4712	0.509525449494429	0.509525449494429	0.000000000000000	0.509525449494429	0.000000000000000
0.6283	0.726542528005361	0.726542528005361	0.000000000000000	0.726542528005361	0.000000000000000
0.7854	1.000000000000000	1.000000000000000	0.000000000000000	1.000000000000000	0.000000000000000
0.9425	1.376381920471170	1.376381920471170	0.000000000000000	1.376381920471170	0.000000000000000
1.0996	1.962610505505150	1.962610505505150	0.000000000000000	1.962610505505150	0.000000000000000
1.2566	3.077683537175250	3.077683537175250	0.000000000000000	3.077683537175250	0.000000000000000
1.4137	6.313751514675040	6.313751514675040	0.000000000000000	6.313751514675040	0.000000000000000
1.5708					
1.7279	-6.313751514675050	-6.313751514674920	-0.000000000000131	-6.313751514675050	0.000000000000000
1.8850	-3.077683537175250	-3.077683537175220	-0.000000000000033	-3.077683537175260	0.000000000000000
2.0420	-1.962610505505150	-1.962610505505140	-0.000000000000015	-1.962610505505150	0.000000000000000
2.1991	-1.376381920471170	-1.376381920471160	-0.000000000000010	-1.376381920471170	0.000000000000000
2.3562	-1.000000000000000	-0.999999999999994	-0.000000000000006	-1.000000000000000	0.000000000000000
2.5133	-0.726542528005361	-0.726542528005356	-0.000000000000005	-0.726542528005361	0.000000000000000
2.6704	-0.509525449494429	-0.509525449494425	-0.000000000000004	-0.509525449494429	0.000000000000000
2.8274	-0.324919696232906	-0.324919696232903	-0.000000000000004	-0.324919696232906	0.000000000000000
2.9845	-0.158384440324536	-0.158384440324533	-0.000000000000003	-0.158384440324536	0.000000000000000
3.1416	0.000000000000000	0.000000000000003	-0.000000000000003	0.000000000000000	0.000000000000000

E. 指数（exp）

【連分数】 $e^x = 1 + 2x/(2 - x + x^2/(6 + x^2/(10 + \cdots + x^2/(2(2n+1) - \cdots$

ここで $e^x = 1 + 2x/(2 - x +$ までは規則的ではありません。この部分以降が規則的な繰返しになっていることに注意してください。

【計算上の補足】 指数の計算は，マクローリン展開でも十分収束が早いので，特に連分数展開である必要はありませんが，ここではあえて連分数で計算する例を示します。

あらかじめ $-1 \le x < 1$ となるよう調整する必要があります。また $x \le 0$ のとき，結果の逆数をとればよいので，簡単化のために絶対値での指数を求めます。

$x \ge 1$ のとき，次のように考えます。

$$e^x = e^n e^{x-n}$$

すなわち，与えられた x を超えない整数 n を見つけ， e^{x-n} を計算し，e の整数乗 e^n を乗じます。しかし，通常のコンピュータでは，機械語として乗算は用意されますが，べき乗の命令は皆無です。一方，言語処理系では，一般にべき乗を計算する必要がありますので，該当する計算機で初めて言語処理系を作る場合，これをソフトウェアで実現する必要があります。

そこで，乗算を繰り返すことになりますが，n が大きい場合，繰返し回数が多く，非常に時間がかかります。ですから，次のように $\log_2 n$ 回のオーダの乗算で済ます方法が一般に用いられます。

```
Public Function myPower(X, N)
 abN = Abs(N): P = 1
 Do While abN > 0
   If (abN Mod 2) = 1 Then P = P * X
   abN = abN ¥ 2: X = X * X
 Loop
 myPower = P
 If N < 0 Then myPower = 1 / P
End Function
```

以上のことから，素直にプログラミングした例を示します。なお，論理的には，連分数展開で求まったEに対して，myPower を使って，

```
myExpF = E * myPowor(A, K)
```


としても良さそうですが，E と A^K の大きさが違いすぎるので，計算誤差が大きくなります。

【プログラム例 1】

```
 Public Function myExpF(X)
  N = 100: XX = Abs(X): K = Fix(XX): XX = XX - K
  A = XX * XX: E = 0
  For i = N To 1 Step -1
    E = A / (2 * (2 * i + 1) + E)
  Next
  E = 1 + 2 * XX / (2 - XX + E)
  A = 2.71828182845905
  Do While K > 0
    If (K Mod 2) = 1 Then E = E * A
    K = K ¥ 2: A = A * A
  Loop
  myExpF = E
  If X < 0 Then myExpF = 1 / E
 End Function
```

対数のところで示したように，*e* の値が近似値であることによる誤差が生じます。対数の場合と同じ方法で改良します。

【プログラム例 2】

```
Public Function myExpF2(X)
   Alf = 0.523536028747135: P = Alf / (1 - Alf): PP = 1 + P
    N = 100: XX = Abs(X): K = Fix(XX): XX = XX - K
    A = XX * XX: E = 0
    For i = N To 1 Step -1
      E = A / (2 * (2 * i + 1) + E)
    Next
    E = 1 + 2 * XX / (2 - XX + E)
    A1 = 2.71828182845904: A2 = 2.71828182845905
    Do While K > 0
      If (K Mod 2) = 1 Then E = E * (A1 + P * A2) / PP
      K = K ¥ 2: A1 = A1 * A1: A2 = A2 * A2
    Loop
    myExpF2 = E
    If X < 0 Then myExpF2 = 1 / E
End Function
```

以下に，myExpF と myExpF2 の比較を示します。当然のことながら

myExpF2 の方が良い結果となっています。

x	ExcelのEXP(x)	myExpF(x)	ExcelのEXP(x)-myExpF(x	myExpF2(x)	ExcelのEXP(x)-myExpF2(x)
-2.0	0.1353352832366130	0.1353352832366120	0.0000000000000005	0.1353352832366130	0.0000000000000000
-1.8	0.1652988882215870	0.1652988882215860	0.0000000000000003	0.1652988882215870	0.0000000000000000
-1.6	0.2018965179946550	0.2018965179946550	0.0000000000000003	0.2018965179946560	0.0000000000000000
-1.4	0.2465969639416060	0.2465969639416060	0.0000000000000004	0.2465969639416070	0.0000000000000000
-1.2	0.3011942119122020	0.3011942119122020	0.0000000000000005	0.3011942119122020	0.0000000000000000
-1.0	0.3678794411714420	0.3678794411714420	0.0000000000000006	0.3678794411714420	0.0000000000000000
-0.8	0.4493289641172210	0.4493289641172210	0.0000000000000000	0.4493289641172220	0.0000000000000000
-0.6	0.5488116360940260	0.5488116360940260	0.0000000000000000	0.5488116360940260	0.0000000000000000
-0.4	0.6703200460356390	0.6703200460356390	0.0000000000000000	0.6703200460356390	0.0000000000000000
-0.2	0.8187307530779820	0.8187307530779820	0.0000000000000000	0.8187307530779820	0.0000000000000000
0.0	1.0000000000000000	1.0000000000000000	0.0000000000000000	1.0000000000000000	0.0000000000000000
0.2	1.2214027581601700	1.2214027581601700	0.0000000000000000	1.2214027581601700	0.0000000000000000
0.4	1.4918246976412700	1.4918246976412700	0.0000000000000000	1.4918246976412700	0.0000000000000000
0.6	1.8221188003905100	1.8221188003905100	0.0000000000000000	1.8221188003905100	0.0000000000000000
0.8	2.2255409284924700	2.2255409284924700	0.0000000000000000	2.2255409284924700	0.0000000000000000
1.0	2.7182818284590400	2.7182818284590400	0.0000000000000000	2.7182818284590400	0.0000000000000000
1.2	3.3201169227365500	3.3201169227365500	-0.0000000000000058	3.3201169227365500	0.0000000000000000
1.4	4.0551999668446700	4.0551999668446800	-0.0000000000000071	4.0551999668446700	0.0000000000000000
1.6	4.9530324243951100	4.9530324243951200	-0.0000000000000080	4.9530324243951100	0.0000000000000000
1.8	6.0496474644129400	6.0496474644129500	-0.0000000000000098	6.0496474644129400	0.0000000000000000
2.0	7.3890560989306500	7.3890560989306600	-0.0000000000000124	7.3890560989306500	0.0000000000000000
2.2	9.0250134994341200	9.0250134994341500	-0.0000000000000320	9.0250134994341200	0.0000000000000000
2.4	11.0231763806416000	11.0231763806416000	-0.0000000000000391	11.0231763806416000	0.0000000000000000
2.6	13.4637380350017000	13.4637380350017000	-0.0000000000000480	13.4637380350017000	0.0000000000000000
2.8	16.4446467710971000	16.4446467710971000	-0.0000000000000568	16.4446467710971000	0.0000000000000000
3.0	20.0855369231877000	20.0855369231878000	-0.0000000000001066	20.0855369231877000	0.0000000000000000